

Getting Started With Alice: The Basics



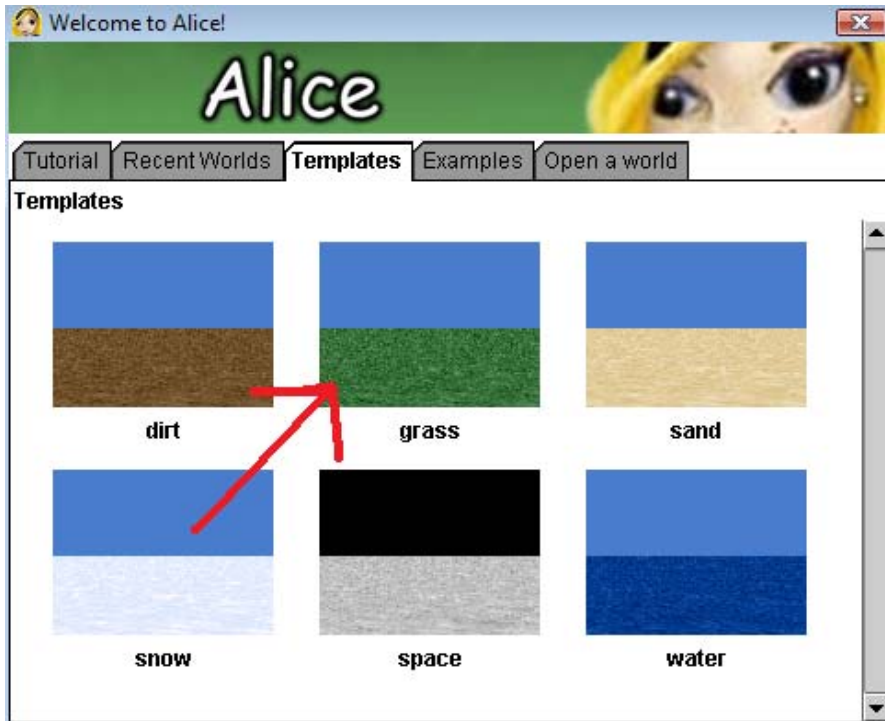
UNT RoboCamp

Alice

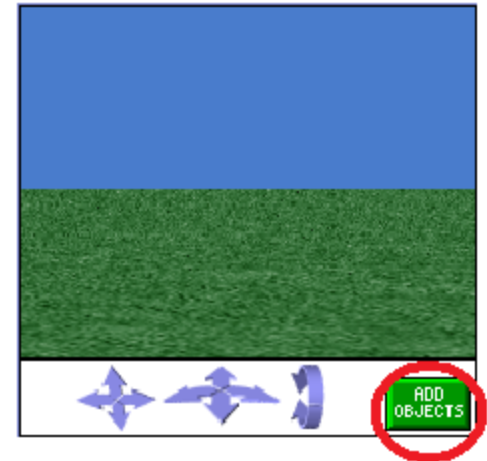
Learning to program: Part 1



Open up Alice, and choose a background for your Alice world. Your world is something you can put **objects** in and make them do things. There are six different background options. Choose **grass**.



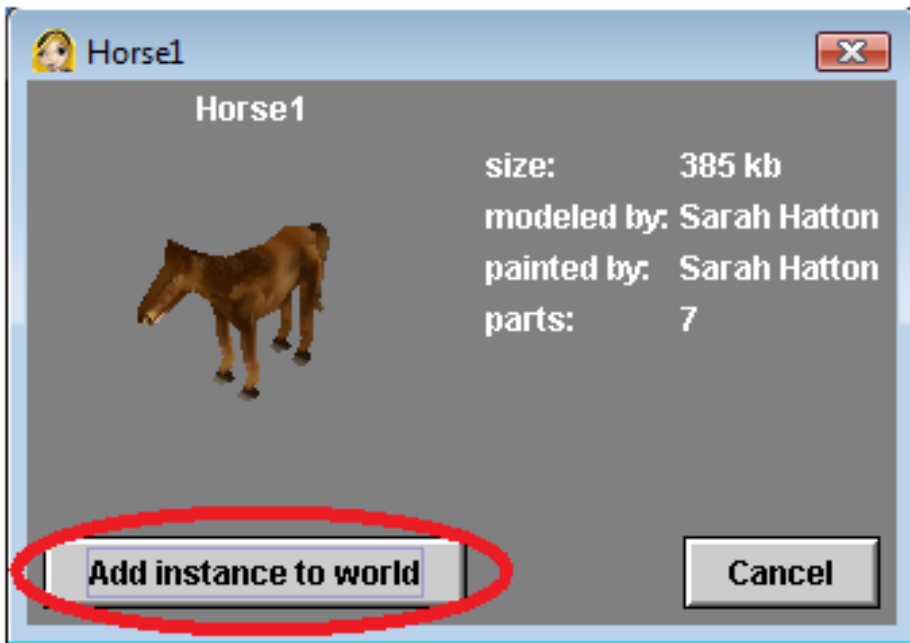
Add an **object** to your Alice world. Alice is full of different kinds of objects to add to your world to make it interesting. Click on the **Add Objects** button:



Click on the **animals** folder of objects:



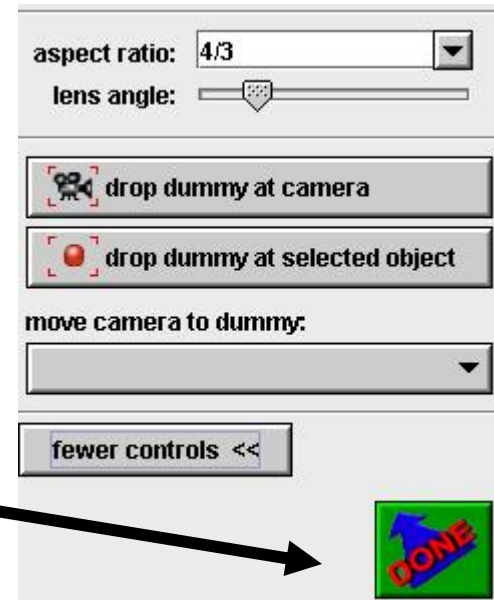
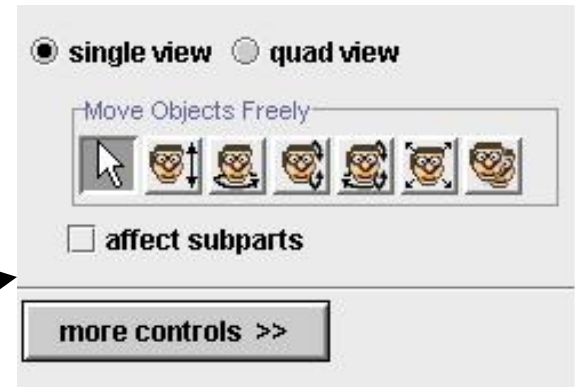
Find the **horse** among the types of objects and click on it. Then click **Add Instance to World**. This is how you add an object to your Alice world.



The horse object will appear in your world.

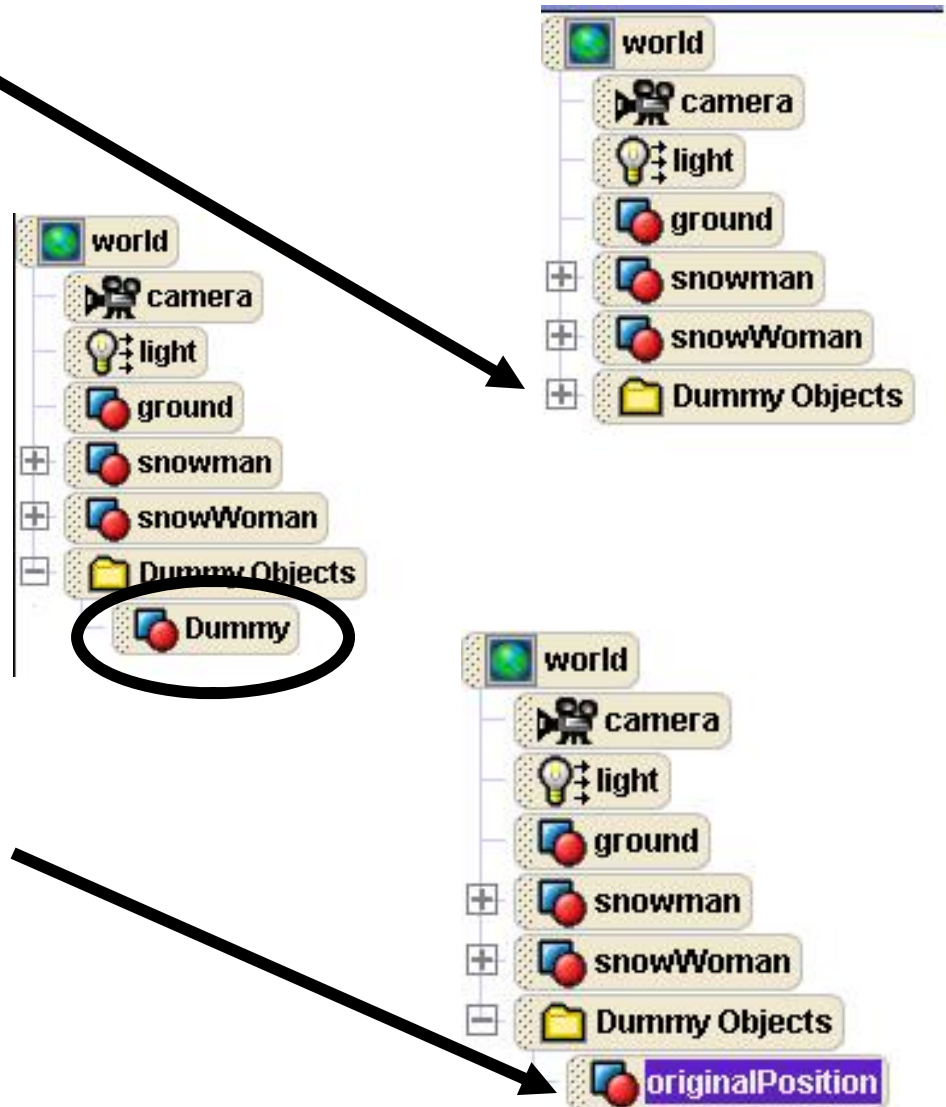
Adding a Dummy Camera

- This is to save current camera position for later
- Click on “more controls”
- Click on “drop dummy at camera”. Just click it once!
- Click the green **Done** arrow on the right hand side of the screen to go back to the main Alice window.



Adding a Dummy Camera (cont)

- A folder of Dummy Objects appears
- Click on the “+” by it
- The camera position saved is “Dummy”
- Click on it and select “rename” and rename it to “original position”
- We will use this later...



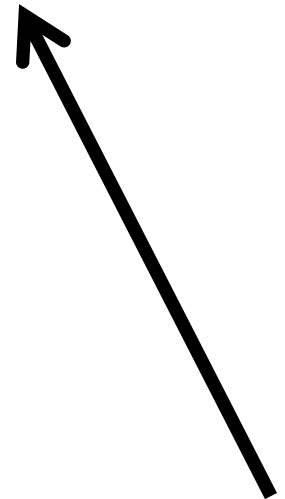
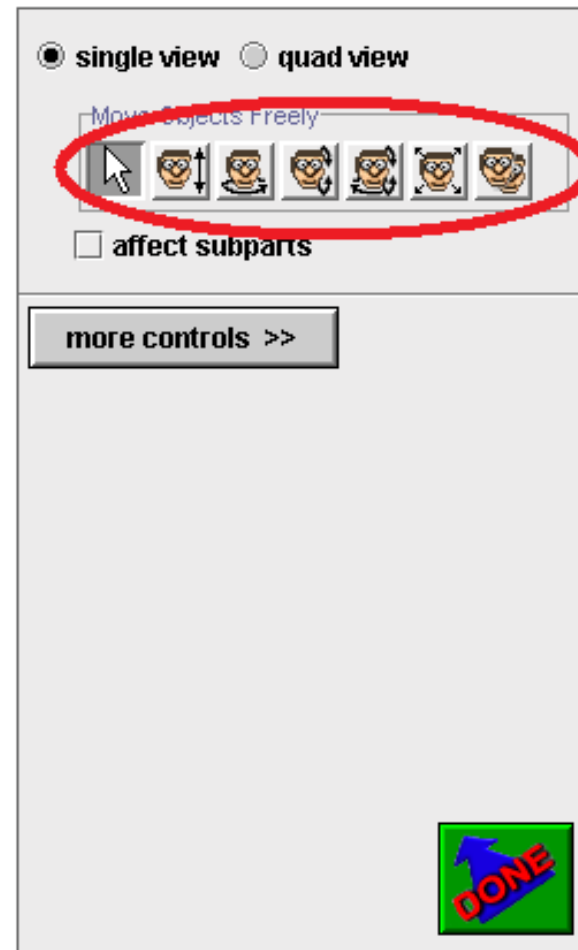
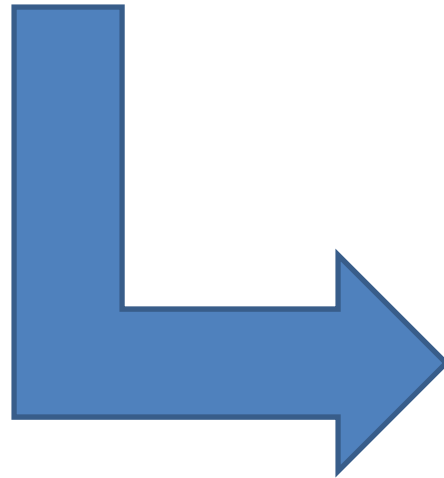
Save World

- Save world

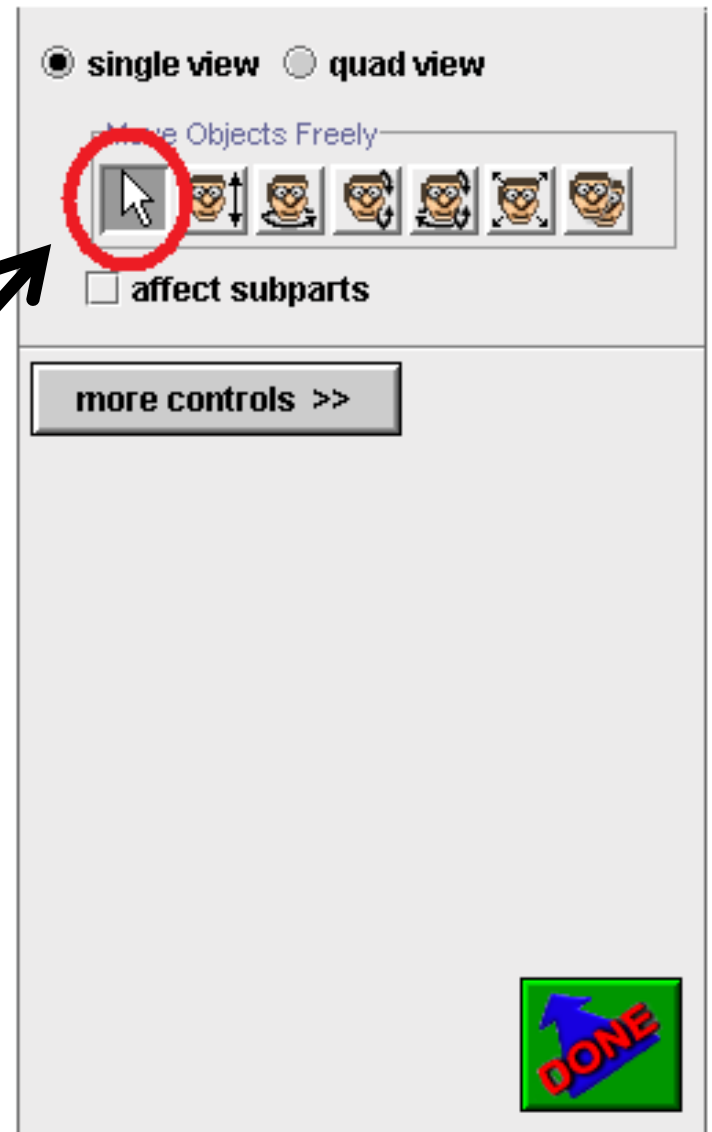
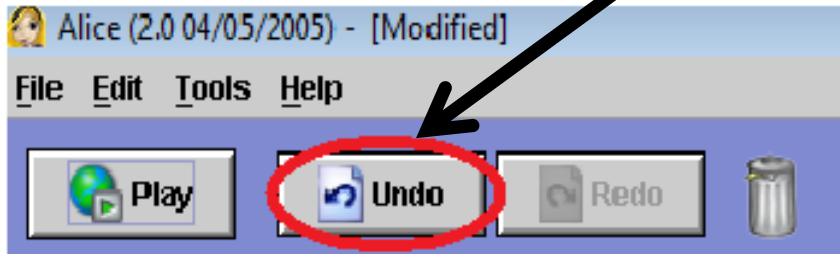


- Type in a name such as: horseWorld, and then “save”
- Alice will remind you to save your work every 15 minutes
- It is always a good idea to create a folder to put all your Alice worlds in.

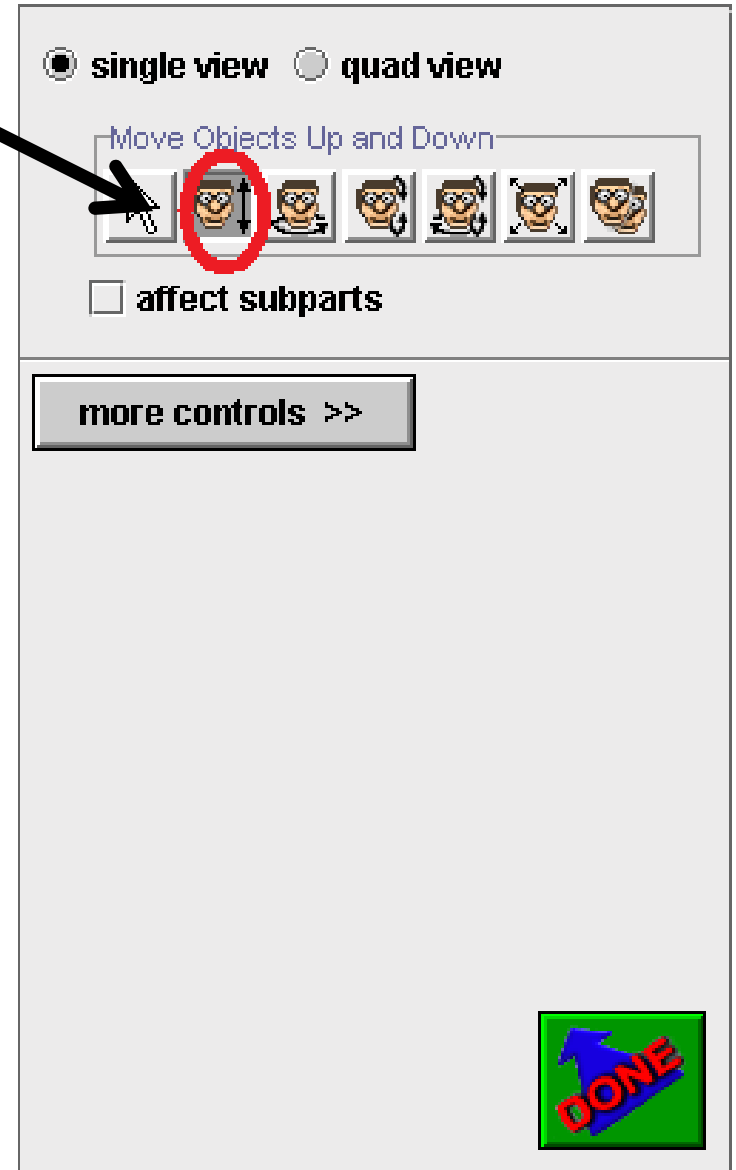
You may see that your horse looks like it's halfway underground. When objects are added, sometimes they appear in strange places. We need to move the horse so he looks like he's on the ground. Click on the **add Objects** button again. We are going to use these buttons that appear to move him around:



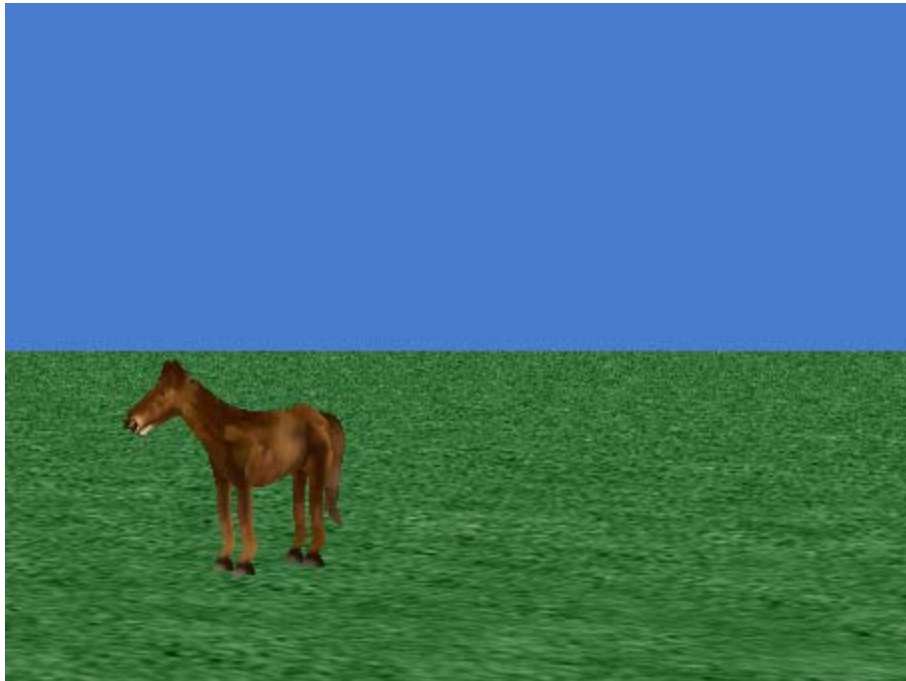
This button will move your horse further back from the camera or closer to the camera, to the right, or to the left. Click on your horse and drag him around. Try moving your horse forwards and backwards. If you don't like the way your horse looks after you move him, click the **Undo** button in the top left corner of the screen.



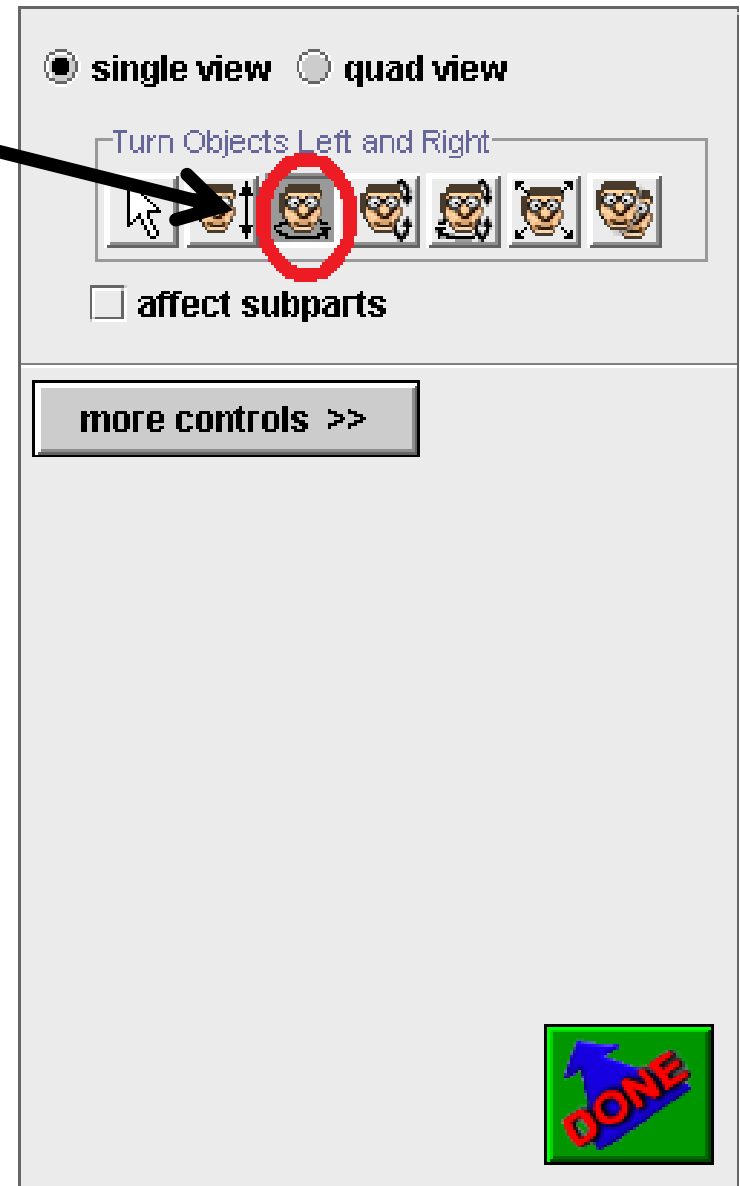
This button moves an object up and down when you click on it and drag it. Move your horse up until he is standing on the ground.



This button turns objects left and right. Use this to turn your horse a little to the left. Use **Undo** if you make a mistake.



Notice that your horse seems to turn around one point. This is your horse's **center**.



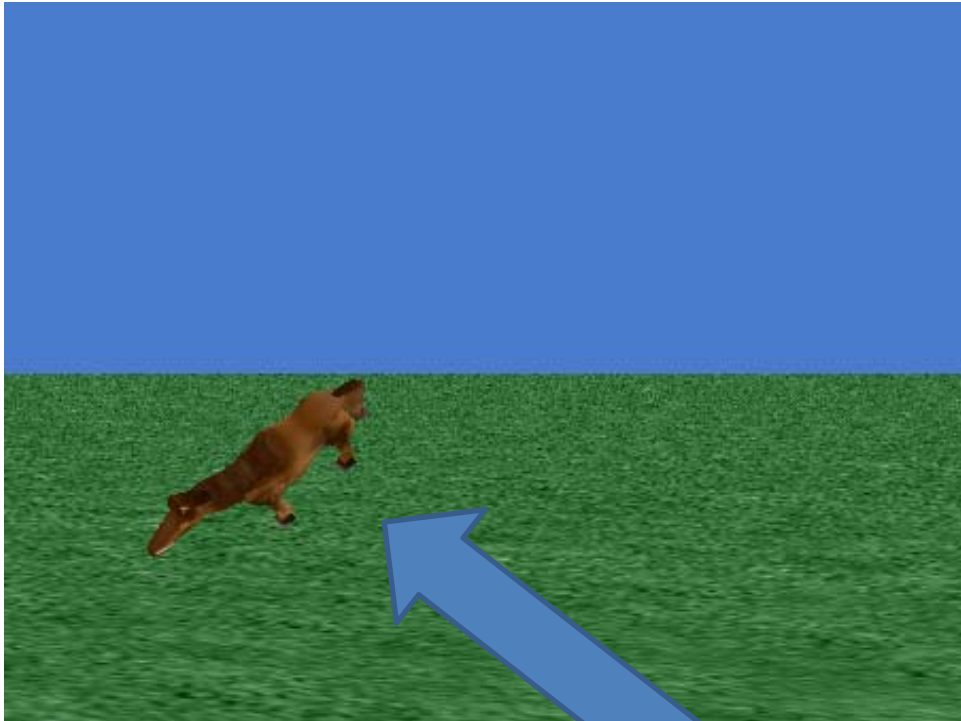
Each object in Alice has a **center**. This is the point on an Alice object that determines where it is in the world, and around which the object will rotate if commanded. Every object's center is different.



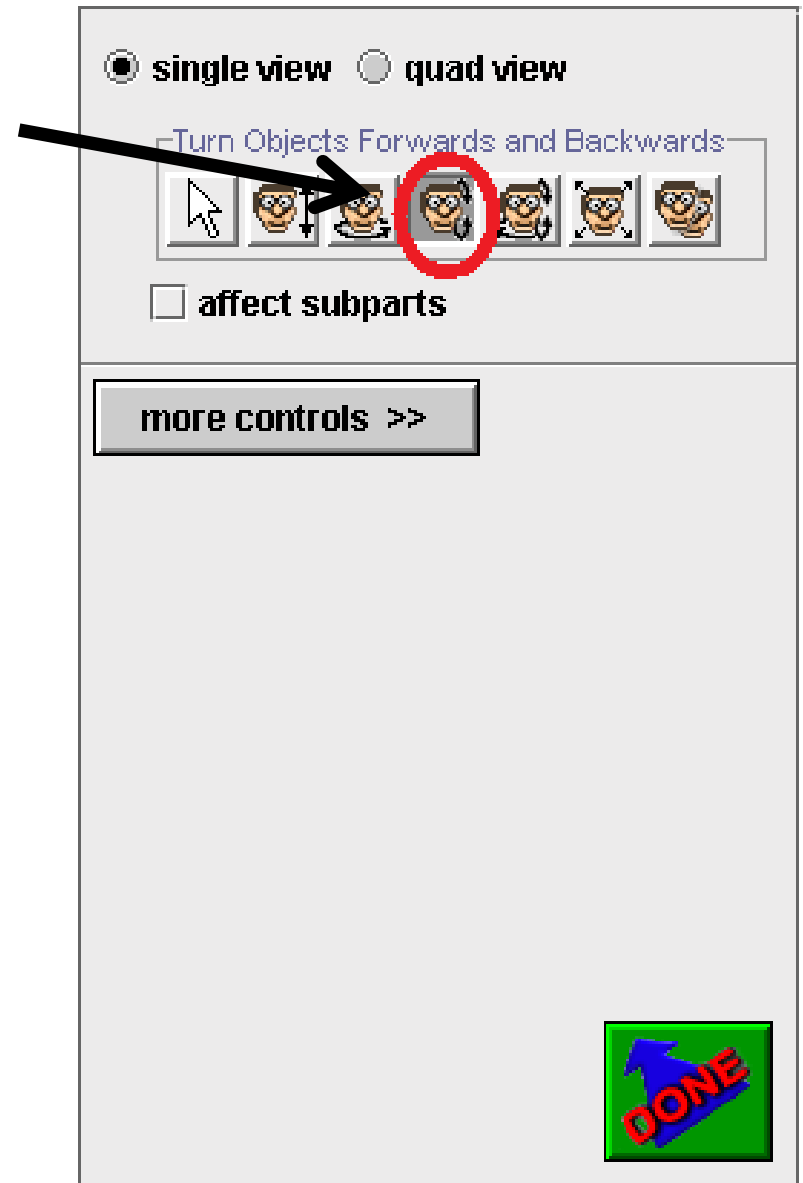
center



This button rotates an object forwards or backwards around its center. Try rotating your horse, and then put it back in its original position using **Undo**.



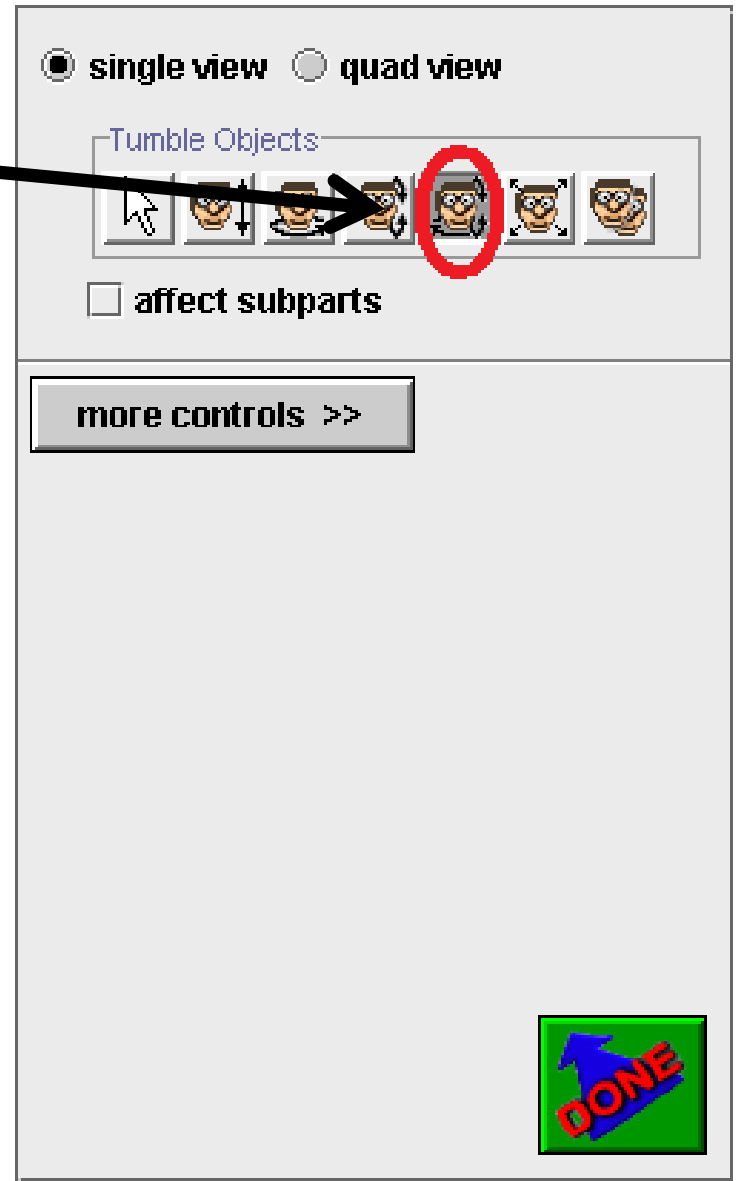
The horse is rotating.



This button tumbles an object in all kinds of crazy directions. Try this button on your horse, and then put it back to its original position using **Undo**.



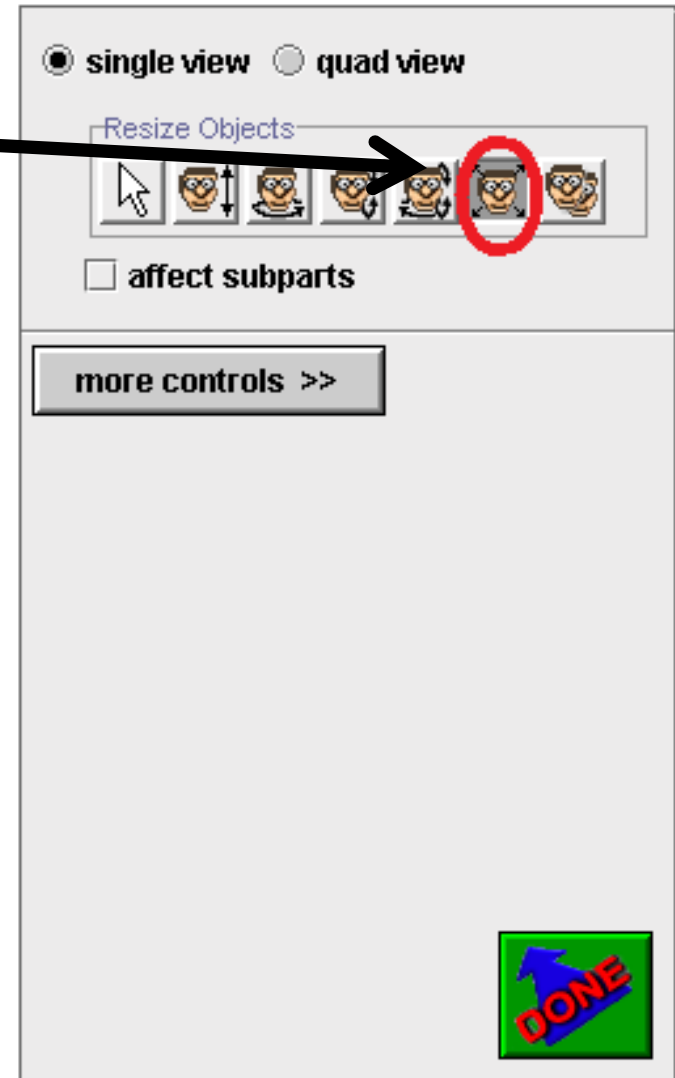
The horse is tumbling!



This button makes an object bigger or smaller. Try re-sizing your horse, so it is a little bigger, and then move him up out of the ground.



Giant horse!



Creating an Event in Alice

- Now we are going to show you how to create an event in Alice.

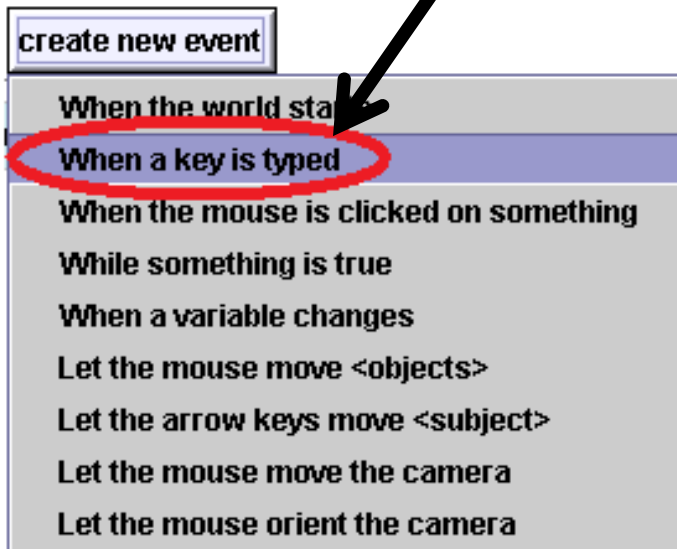
- An event is something that you tell Alice to do after something else happens, such as pressing a key or clicking on something.

In the top right hand corner of the screen is the **Events Editor**. This is where you make events, which are commands that you tell Alice to do when a certain thing happens, such as when a key is pressed, or you click your mouse on something.



Click on the **create new event** button.
Choose the **when a key is typed** option.

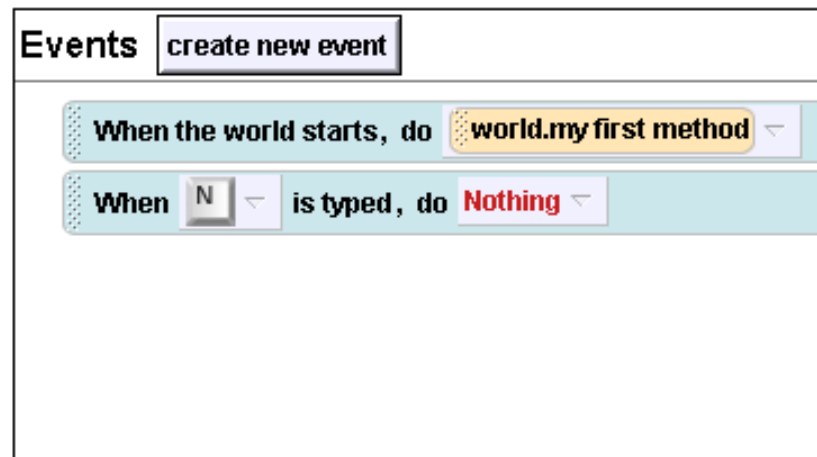
We are going to make it so that when we press the **N** key, our horse will say “NEEEEEEEIGH!”



Your event editor will now look like this:



Click on **any key**, then choose **letters**, then choose **N**. It will now look like this:



Now click on the **horse** in the list of objects, and then click on the **methods** tab. Find **horse say**, and drag it up to the events editor where it now says **Nothing**.

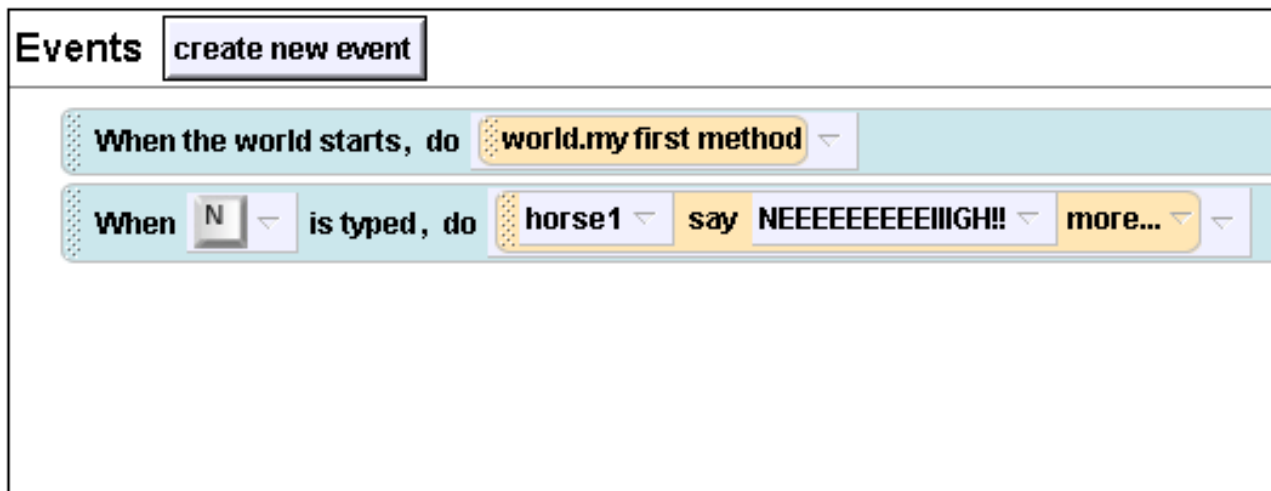
The screenshot displays a game development environment with several panels:

- World Panel (Top Left):** A hierarchical list of objects including 'world', 'camera', 'light', 'ground', 'horse1', and 'Dummy Objects'.
- 3D Viewport (Top Center):** A 3D scene showing a brown horse on a green field under a blue sky, with a yellow wireframe bounding box around it.
- Events Panel (Top Right):** Contains a 'create new event' button and two event triggers:
 - 'When the world starts, do world.my first method'
 - 'When N is typed, do Nothing'
- horse1's details Panel (Bottom Left):** Has tabs for 'properties', 'methods', and 'functions'. The 'methods' tab is active, showing a list of methods for 'horse1': 'move', 'turn', 'roll', 'resize', 'say', 'think', and 'play sound'. The 'say' method is circled in black.
- Method Editor Panel (Bottom Center):** Displays 'world.my first method' with 'No parameters' and 'No variables'. It contains a 'Do Nothing' button.

A large black arrow originates from the circled 'horse1 say' method in the 'horse1's details' panel and points to the 'Nothing' dropdown in the 'When N is typed, do' event trigger in the 'Events' panel.

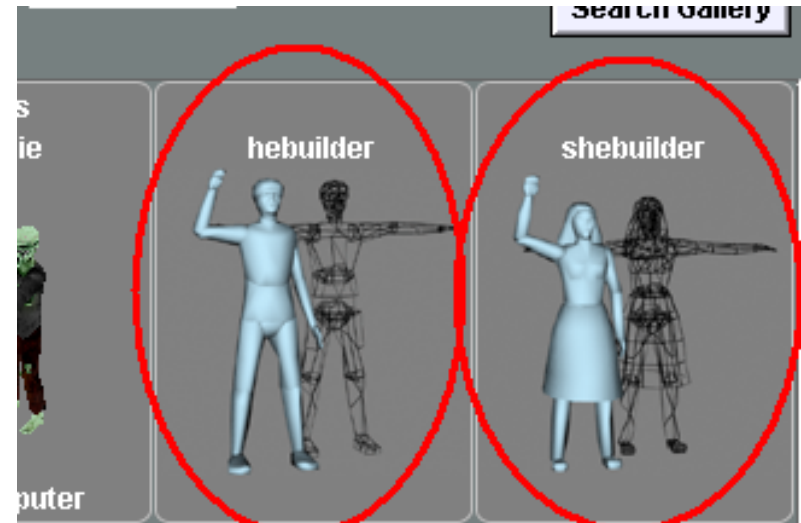
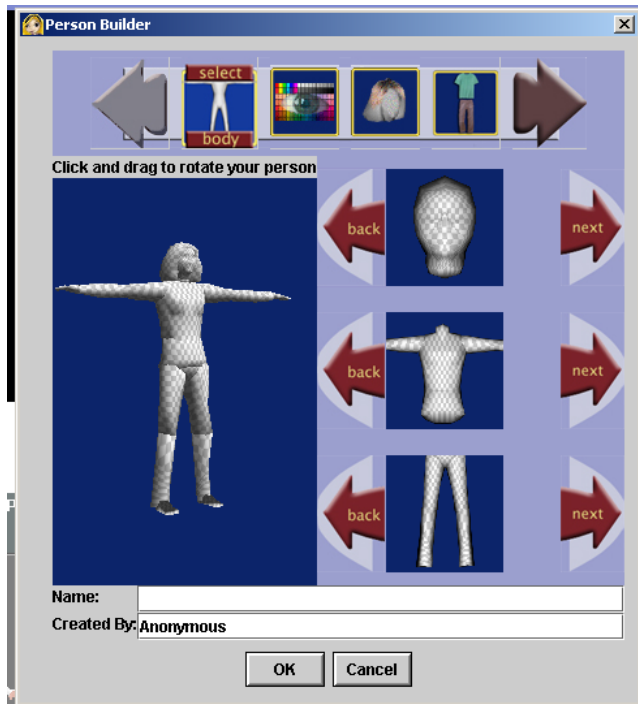
When you drop the **say** command, you will need to type in something for the horse to say, like “NEEEEEEEEEIIIGH!” Your events editor will now look like this:

Play your world, and try pressing **N**. Whenever you press it, your horse should neigh.



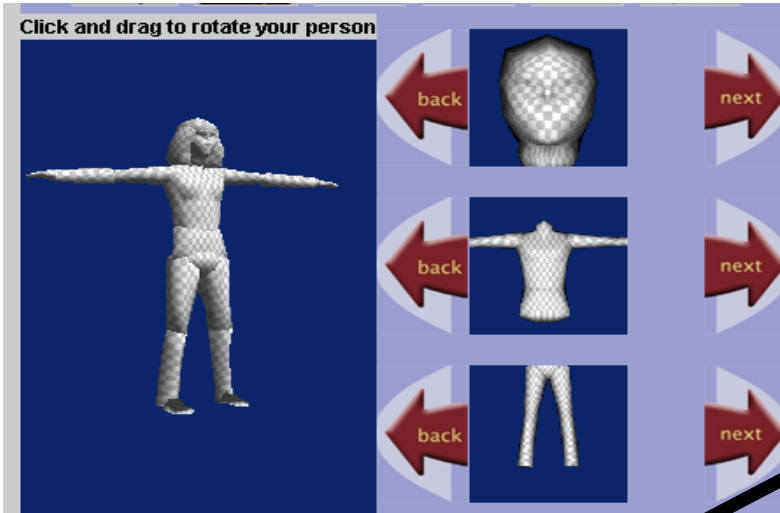
Now we are going to use the he-builder and she-builder objects in Alice.

- The he/builder and she/builder are for creating your own characters in a world.
- You can find it under the “people” folder of your Local Gallery



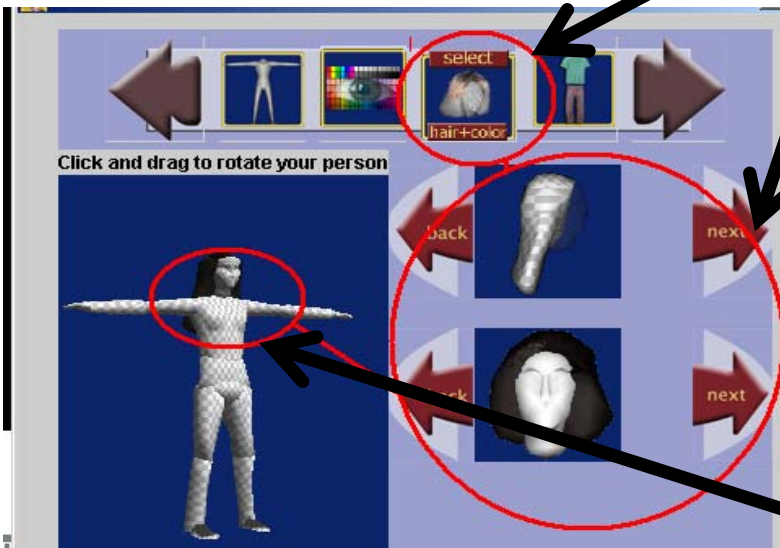
- Lets start with the She/builder
- When you open the she/builder folder, this screen will appear

Getting started



- You can change the leg, upper body and head shape of your person, using the arrows.

- Select the “hair color”. You can change the shape and color of your hair, using arrows.



Choosing the Details



- Now, select the tab labeled “skin/eyes”. You should have three options: skin color, eyes and lips. The skin color is controlled by a dial on the top.
- The eyes and lips can be selected by the arrows.
- All of the changes you make will appear on your person, to the left.

Choosing the Details



- Now, select the tab labeled “skin/eyes”. You should have three options: skin color, eyes and lips. The skin color is controlled by a dial on the top.
- The eyes and lips can be selected by the arrows.
- All of the changes you make will appear on your person, to the left.

Choosing the Details (continued)

- By selecting the “clothing” tab, you can choose shirts, pants and shoes for your person.
- Finally, you can give your person a name at the bottom of the “person builder”.



Now its your turn



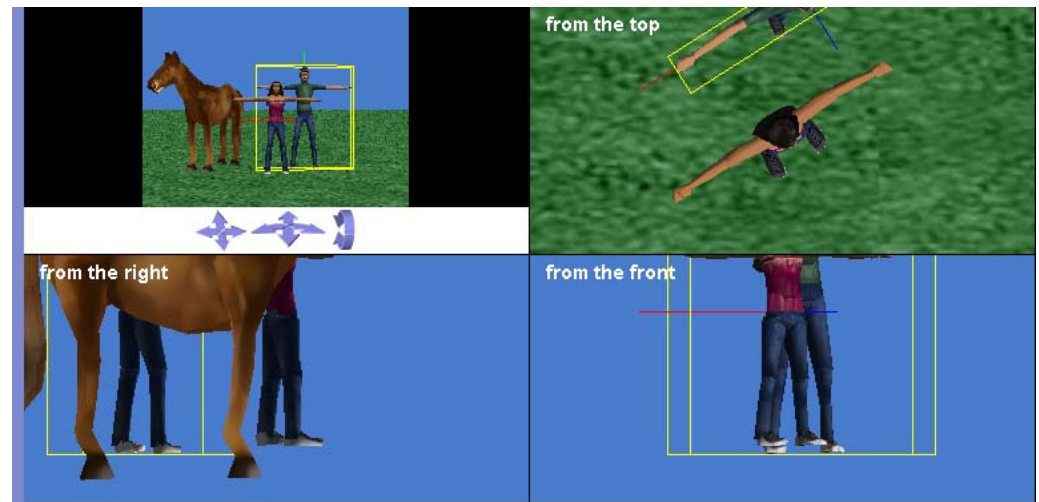
- Try creating another character
- If you want to use the He/builder, it works the same way

- Try experimenting with the buttons that move objects to move your people around in your world.



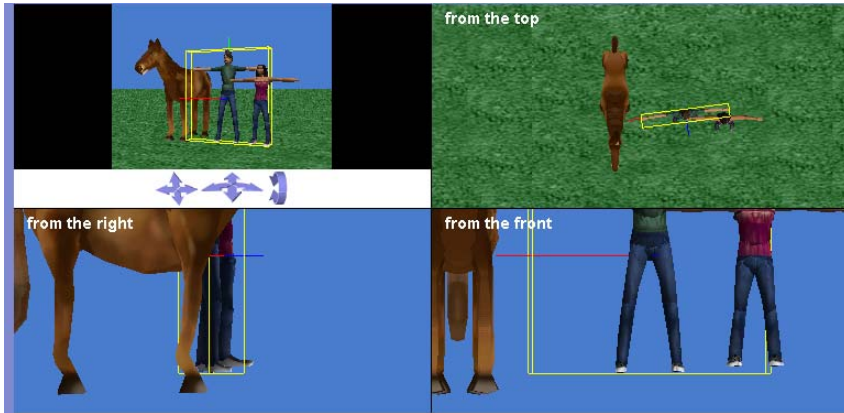
Now we'll use quad view to position our characters. Click the **add Objects** button to get back to the object moving screen.

- Look over on the right side of your screen.
- There should be two types of camera views
- Single view (bottom left) and Quad view (bottom right)



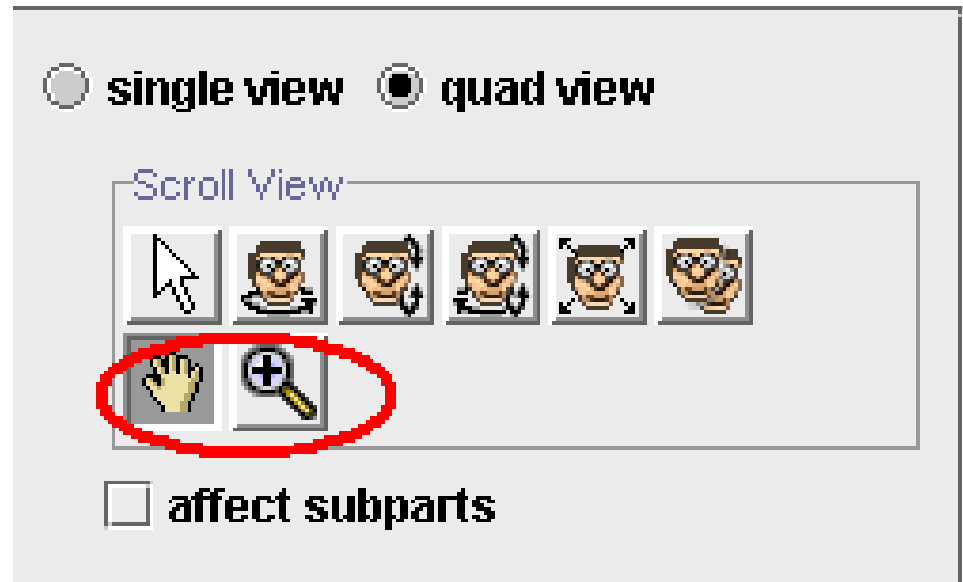
Quad View

- Quad view is good for fixing the positioning of your characters in the world
- Try to position your characters so that they appear in all four screens of your world.
- This helps to make sure that your characters aren't up in the air, or far apart.



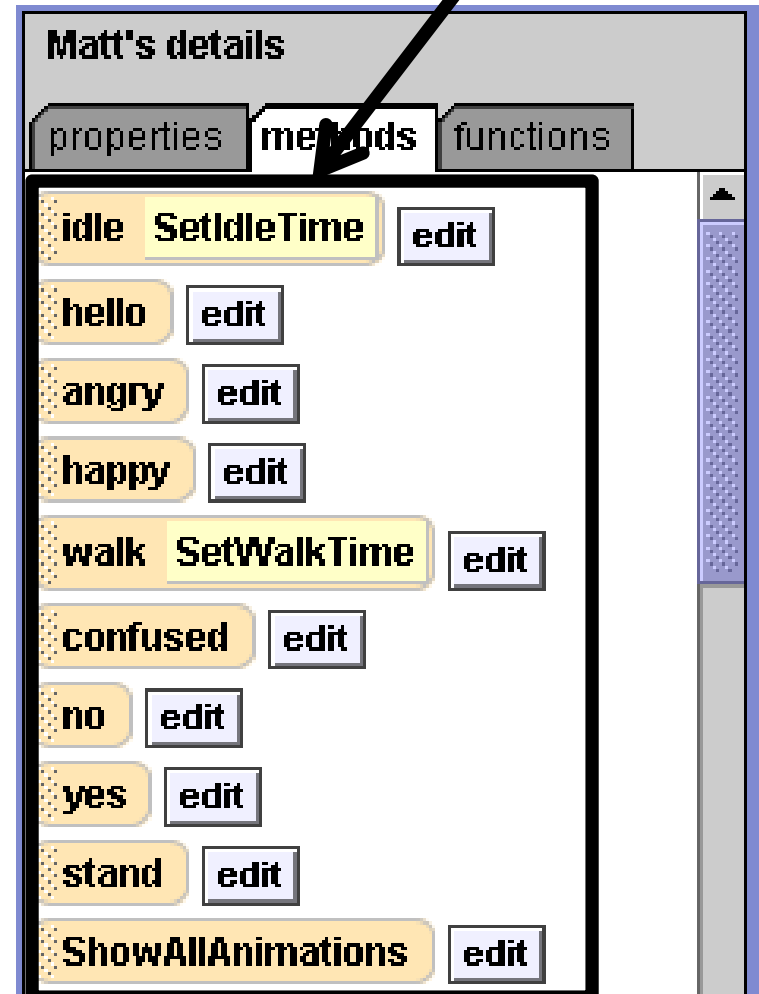
Quad View (Continued)

- Look over on the right, under quad view.
- Note the button that looks like a hand
- This allows you to move around the camera in Quad View
- The Magnifying glass allows you to zoom in while in quad view, to better position your characters
- You'll need to click and drag on your viewing windows when using both of these buttons.
- Try them out now to center your characters in each view.



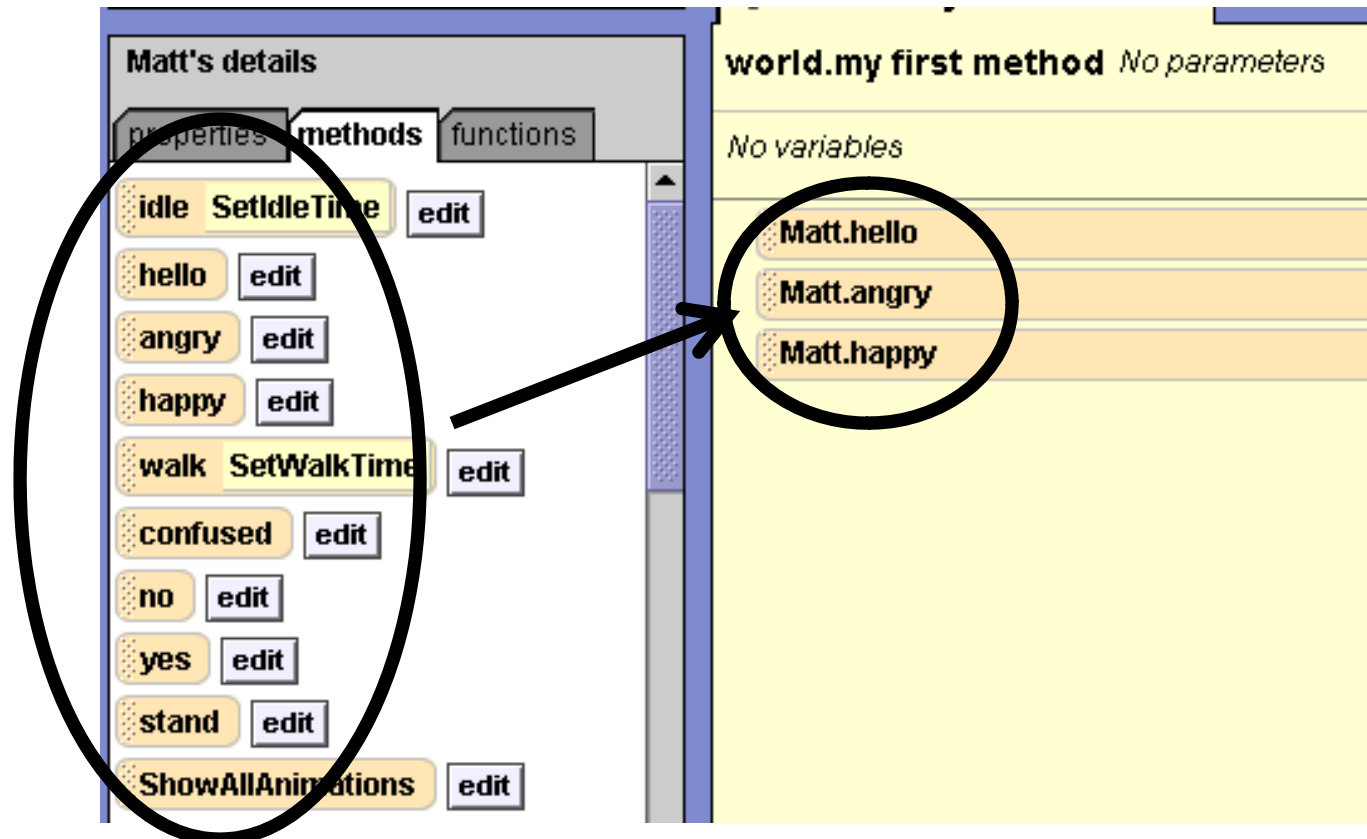
Animating your Characters

Once you have your characters in your world you can start to animate them. Click on one of your characters in the object list. For us it is Matt. Look on the left of your screen under Matt's details. There should be a list of methods. This is the list of actions that your person can already do.



Starting your Own Method

- You can click and drag any of these methods into the main section called “world.my first method”. Try this with a few of them.
- Now click “play” in the top left hand corner.
- Your character should animate however you have coded him.



Move one of your characters around in your world using quad view until they are standing right next to the horse, something like this:



Nice job! Now we are ready to move on to Part 2.

Alice

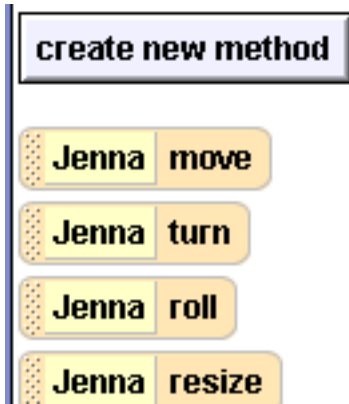
Learning to program: Part Two Writing Your Own Methods



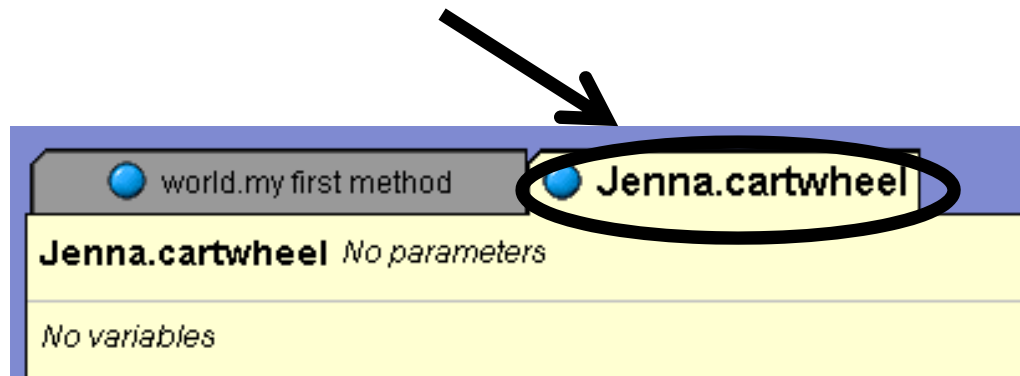
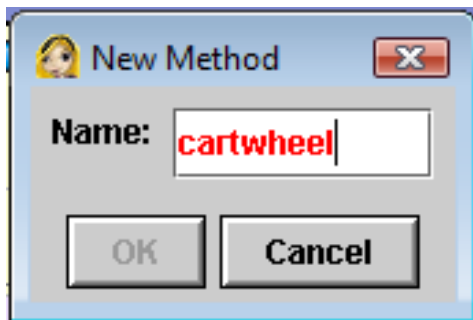
Making Your Own Methods

- What if we want our character to do something that he doesn't already know how to do?
- Now we will write a new method for Jenna to allow her to cartwheel.

Creating Your First Method

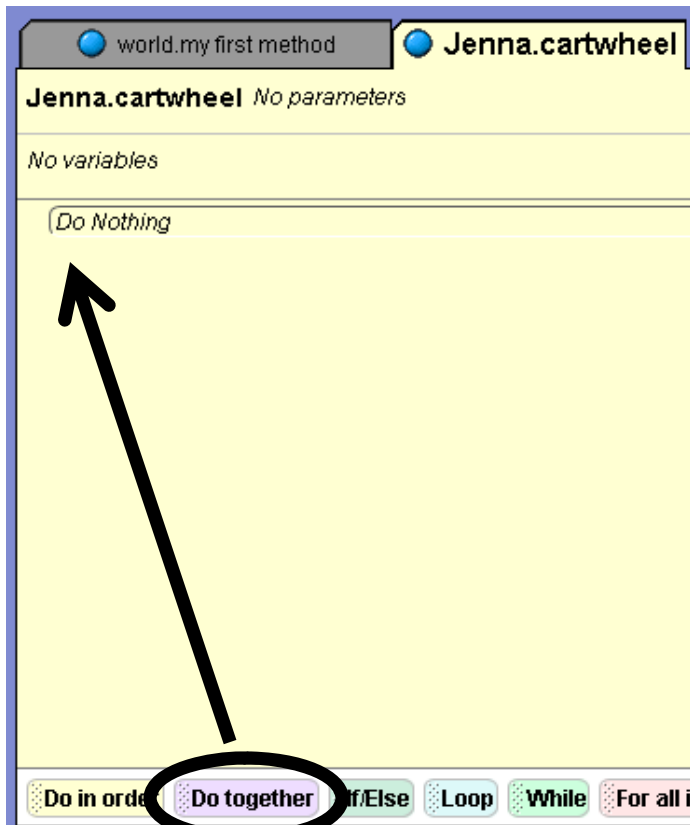


- Click on Jenna's name in the list on the left and scroll down to the area labeled "create new method".
- A gray box will pop up. Label it "cartwheel"
- Now, in the main box there should be a tab named "Jenna.cartwheel"



Creating Your First Method

- We need to code each of the character's body parts individually. Drag up a **Do together** from the bottom of the Alice window and drop it into your cartwheel method.
- Click on the body part that you want to code and look down at the series of commands
- Now look at the code on the next slide and make your code match it.



-Now click on the plus sign beside your character in the object list. The parts should expand out.

Drag and Drop in this code.

The screenshot shows a programming environment with the following components:

- Object Hierarchy (Left):**
 - Jenna
 - LowerBody
 - UpperBody
 - Chest
 - RightUpperArm** (highlighted with a red box)
 - LeftUpperArm
 - Neck
 - horse1
 - Matt
- Stage (Center):** A 3D scene with a horse, a person, and a character named Jenna. A yellow rectangle highlights the character's arms.
- Code Editor (Right):**
 - Event: When the world starts, do world.my first method
 - Function: Jenna.cartwheel
 - Block: Do together
 - Jenna.UpperBody.Chest.RightUpperArm roll left 0.18 revolutions more...
 - Jenna.UpperBody.Chest.LeftUpperArm roll right 0.18 revolutions more...

Arrows indicate the process of dragging the 'roll' method from the 'RightUpperArm' object's methods list into the 'Do together' block in the code editor.

For example, click on **RightUpperArm** to see the methods for just this arm. Then, drag over the **roll** method, and set it to roll **left .18 revolutions**. Repeat this process for **LeftUpperArm**, choosing the opposite direction.

Final code

- Your final code should look like this. You should drag and drop in the rest of it.

Jenna.cartwheel *No parameters*

No variables

☐ Do together

Jenna.UpperBody.Chest.RightUpperArm ▾ roll left ▾ 0.18 revolutions ▾ more... ▾

Jenna.UpperBody.Chest.LeftUpperArm ▾ roll right ▾ 0.18 revolutions ▾ more... ▾

☐ Do together

Jenna ▾ roll left ▾ 1 revolution ▾ more... ▾

☐ Do together

Jenna.UpperBody.Chest.RightUpperArm ▾ roll right ▾ 0.18 revolutions ▾ more... ▾

Jenna.UpperBody.Chest.LeftUpperArm ▾ roll left ▾ 0.18 revolutions ▾ more... ▾

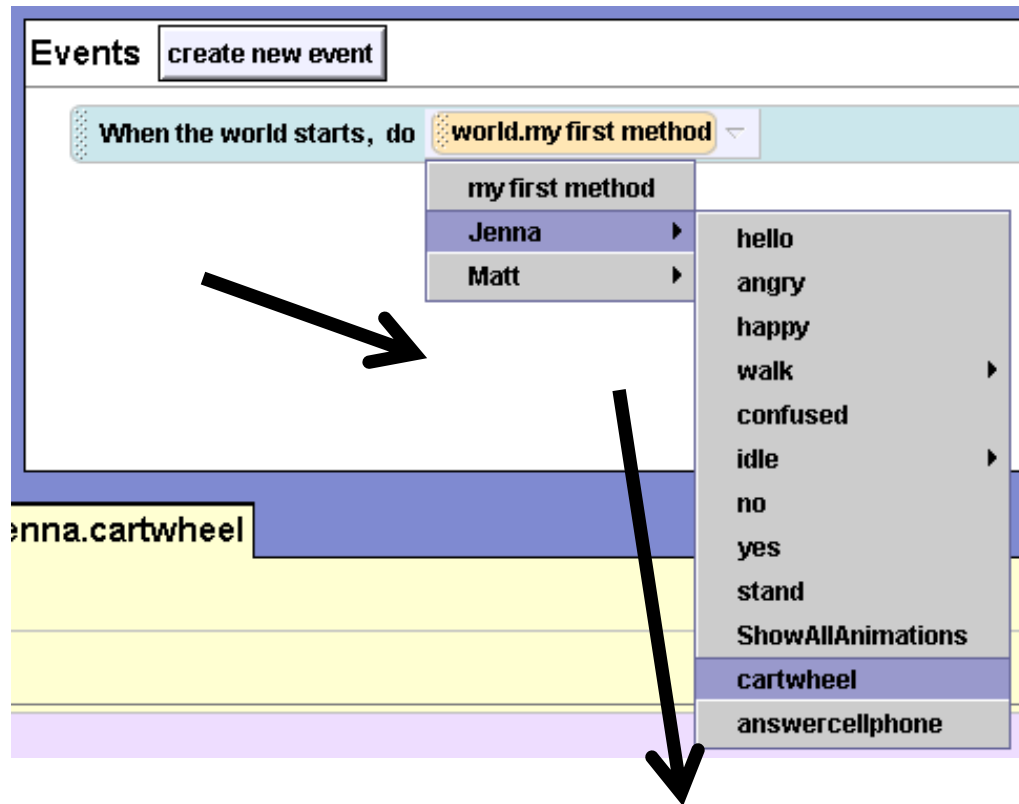
Now click play.



You should notice that nothing happens. Why is this? Look in the top right hand corner of your screen, and find where it says **when the world starts do.**

Notice that it says, do **world.my first method.** The method that you just wrote is called Jenna.cartwheel.

Playing Your New Method



Look at the right corner of your screen. There should be a section called “Events”.

Click on “When the World Starts”

Select “Jenna” and then “cartwheel”.

Now select Play

Your Character should animate into a cartwheel



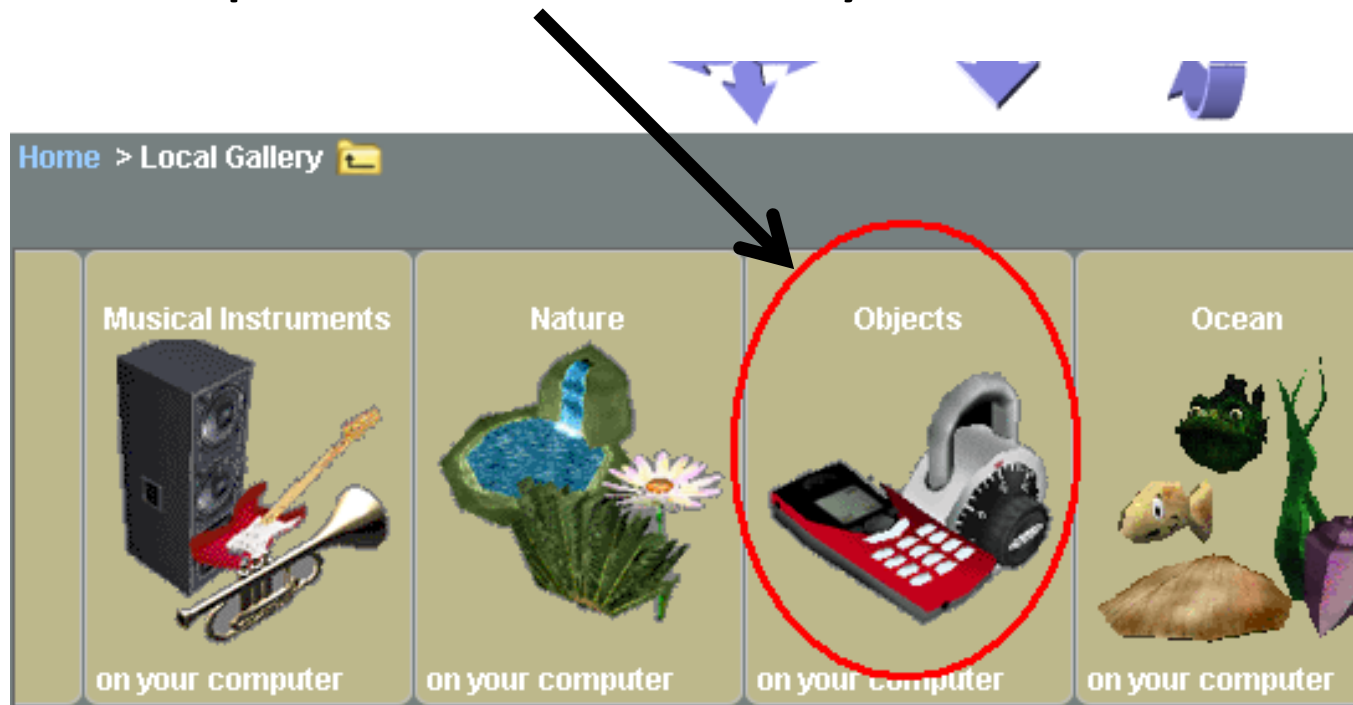
This is a good way to test a method that you have just written.

Methods (Continued)

- Now lets create a slightly more complicated method.
- Click on the “Create New Method” button again and name this one “answercellphone”.

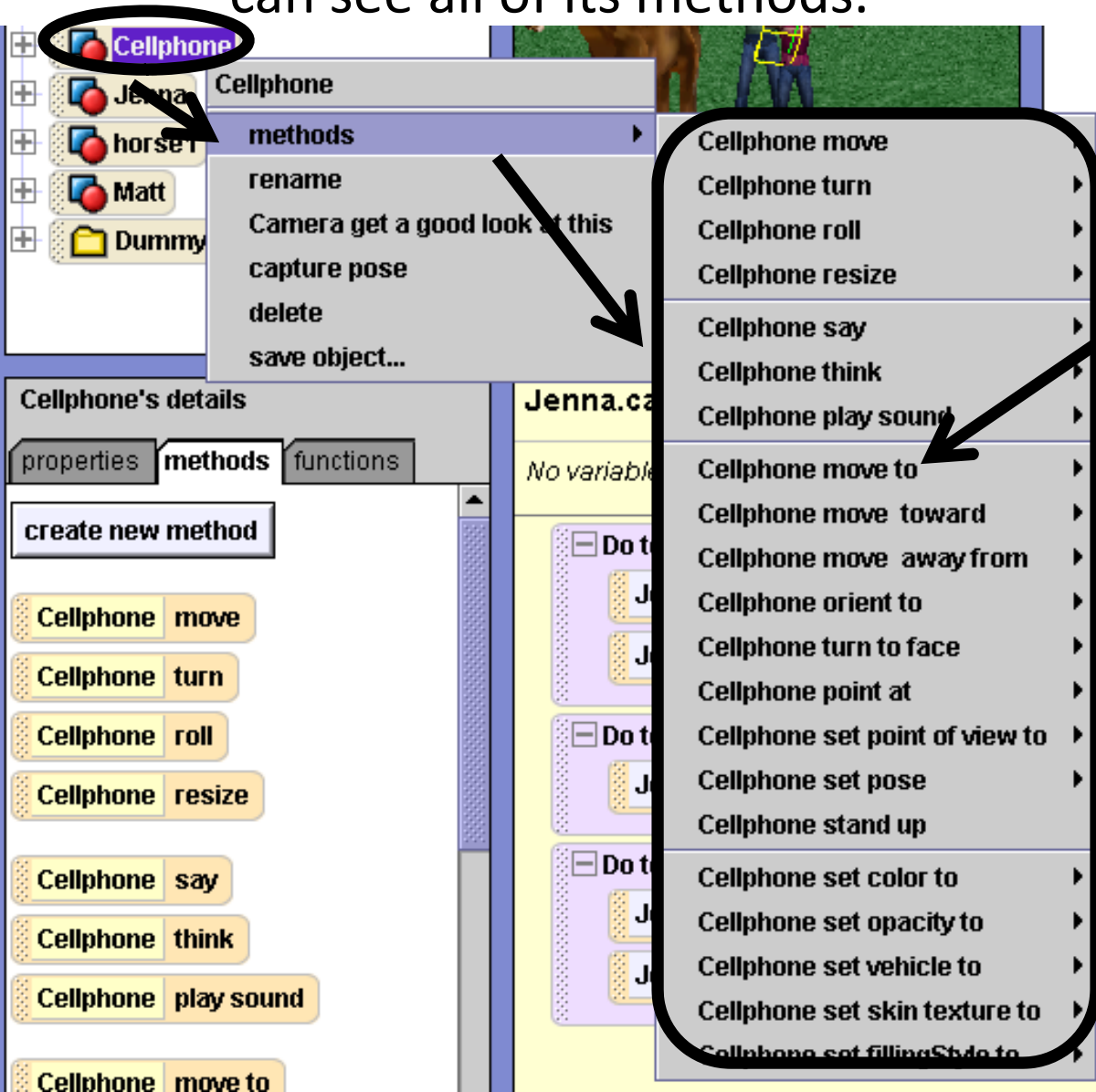
Methods (Continued)

- Click on “Add Objects” and scroll down to the “objects” folder. There should be an object labeled “cell phone”. Add it to your world.



Positioning the Phone

- If you right click on the cell phone in the object tree you can see all of its methods.



-Select **move to**, and have the phone move to Jenna.

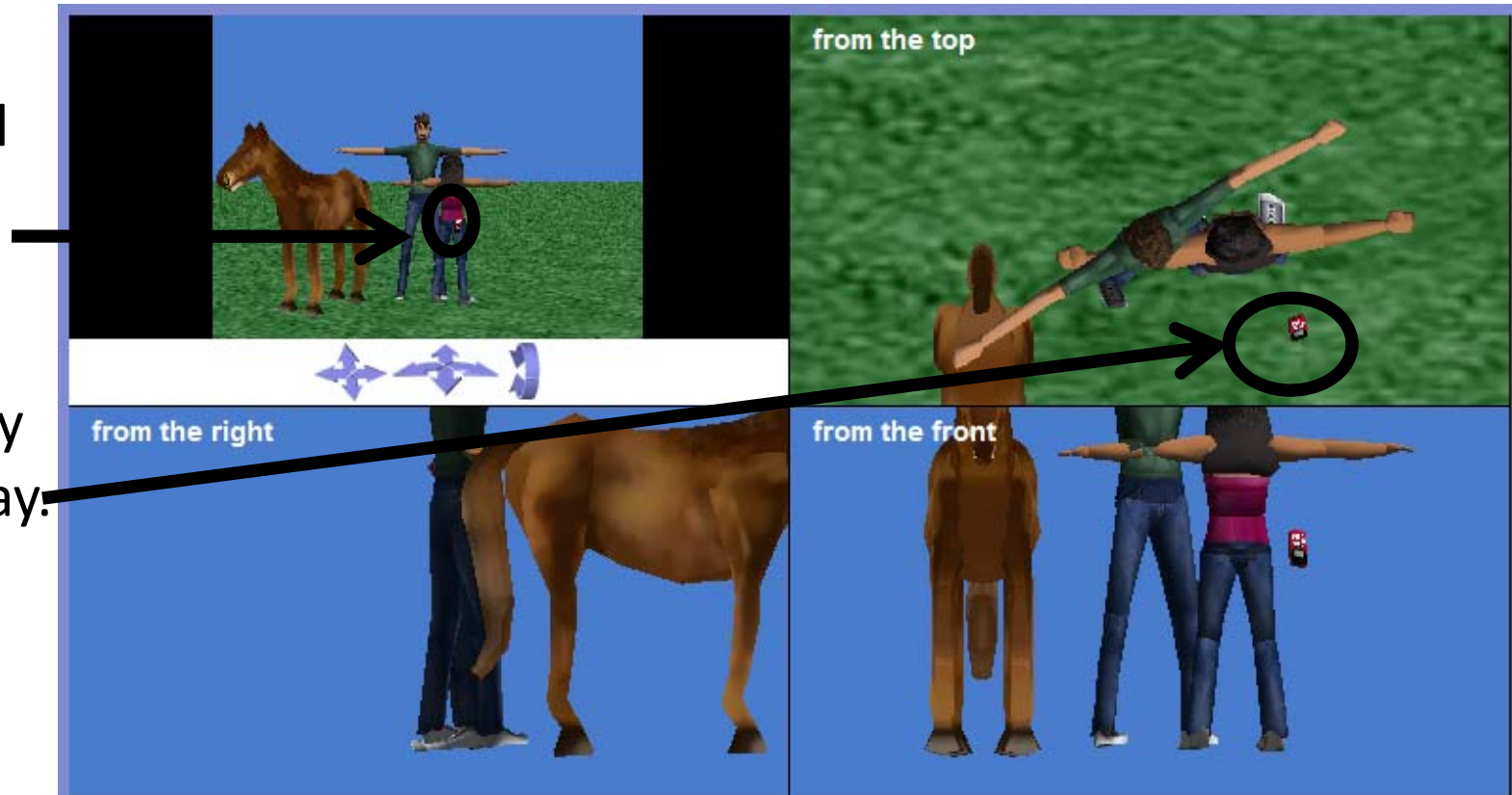
-This will simply position your object in your viewing screen. It will not change your code that you have written.

Positioning the Object

- Now use the object moving buttons to place the phone near the back pocket of your character.
- You can turn your character around to do this.

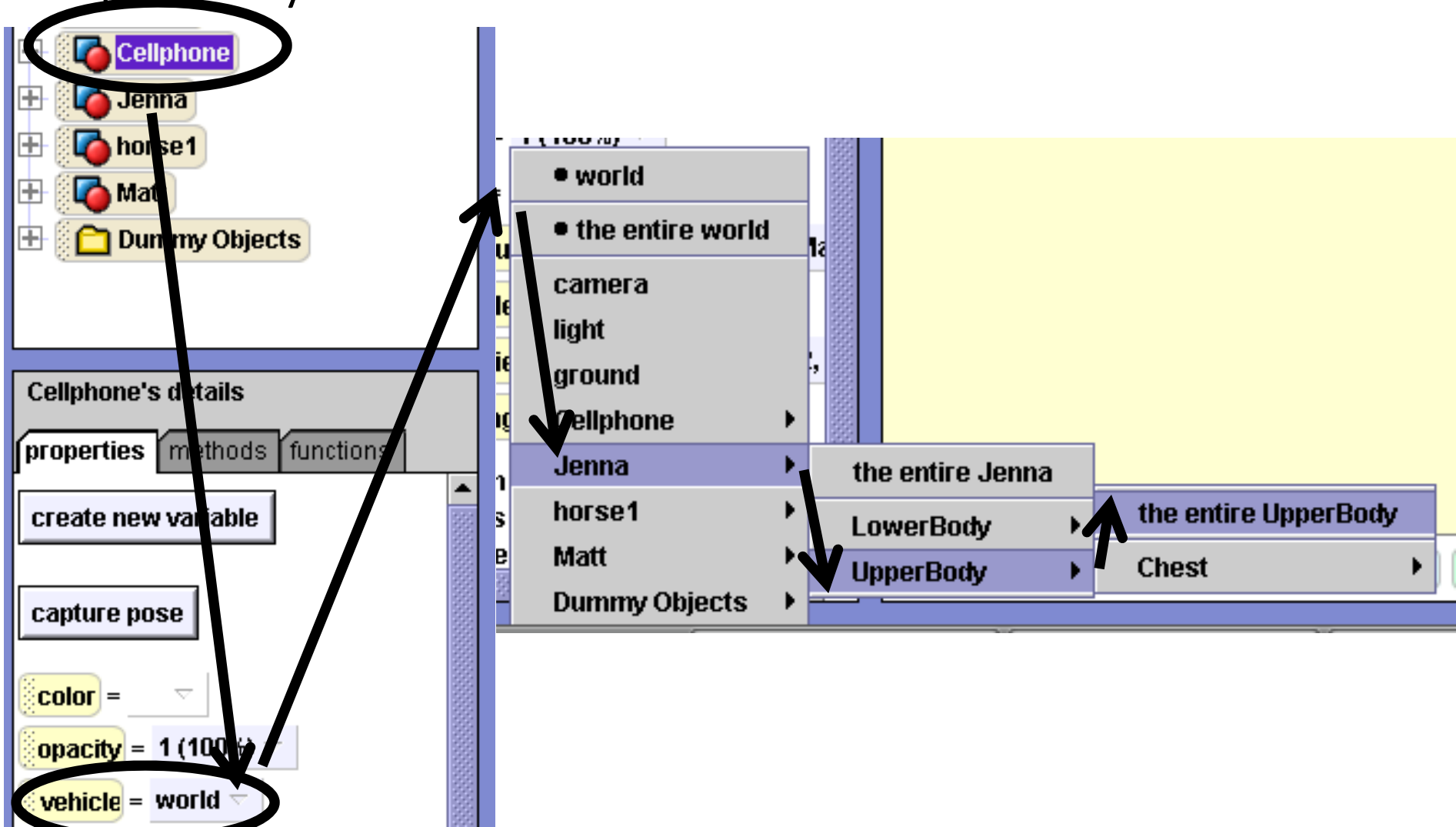
-You should also try using quad view to make sure that the cellphone is very close to her pants, because it is hard to tell from the normal view.

Looks normal from here, but its actually far away.



Glue Cellphone to Pocket

- Now select the properties tab of “cell phone” and scroll down to the command called “vehicle”. Click on the down arrow next to the Vehicle button and set the Cell phone as a vehicle to the character’s upper body.



Vehicle Property

- The vehicle property basically allows you to glue one object to another object.
- If you set the cell phone as a vehicle to the character, you can move your character around and the cell phone will follow. Turn your character around with your object moving buttons to see the cellphone move with him or her.
- Now we can code the character to take the cell phone out of her back pocket.

Method “answercellphone”

- The first part is simply to code your characters arm to reach down and grab her cell phone

[-] Do together

Jenna.UpperBody.Chest.RightUpperArm ▾

roll right ▾

0.18 revolutions ▾

more... ▾

Jenna.UpperBody.Chest.RightUpperArm.RightForearm ▾

roll right ▾

0.18 revolutions ▾

more... ▾

- Now go back to your vehicle property and drag it into the method to set the cell phone as a vehicle to your character's hand. The cell phone is now glued to your character's hand.

Cellphone ▾

set vehicle to

Jenna.UpperBody.Chest.RightUpperArm.RightForearm.RightHand.RightFingers ▾

more... ▾

Method “answercellphone”

- Now code the arm to come up to your characters ear and have your character say “hello”.

☐ Do together

Jenna.UpperBody.Chest.RightUpperArm ▾ roll left ▾ 0.18 revolutions ▾ more... ▾

Jenna.UpperBody.Chest.RightUpperArm.RightForearm ▾ roll left ▾ 0.5 revolutions ▾ more... ▾

☐ Do together

Jenna ▾ say hello ▾ *duration* = 4 seconds ▾ more... ▾

Method “answercellphone”

Here is the completed method.

Jenna.answercellphone *No parameters*

No variables

[-] Do together

Jenna.UpperBody.Chest.RightUpperArm ▾ roll right ▾ 0.18 revolutions ▾ more... ▾

Jenna.UpperBody.Chest.RightUpperArm.RightForearm ▾ roll right ▾ 0.18 revolutions ▾ more... ▾

Cellphone ▾ set vehicle to Jenna.UpperBody.Chest.RightUpperArm.RightForearm.RightHand.RightFingers ▾ more... ▾

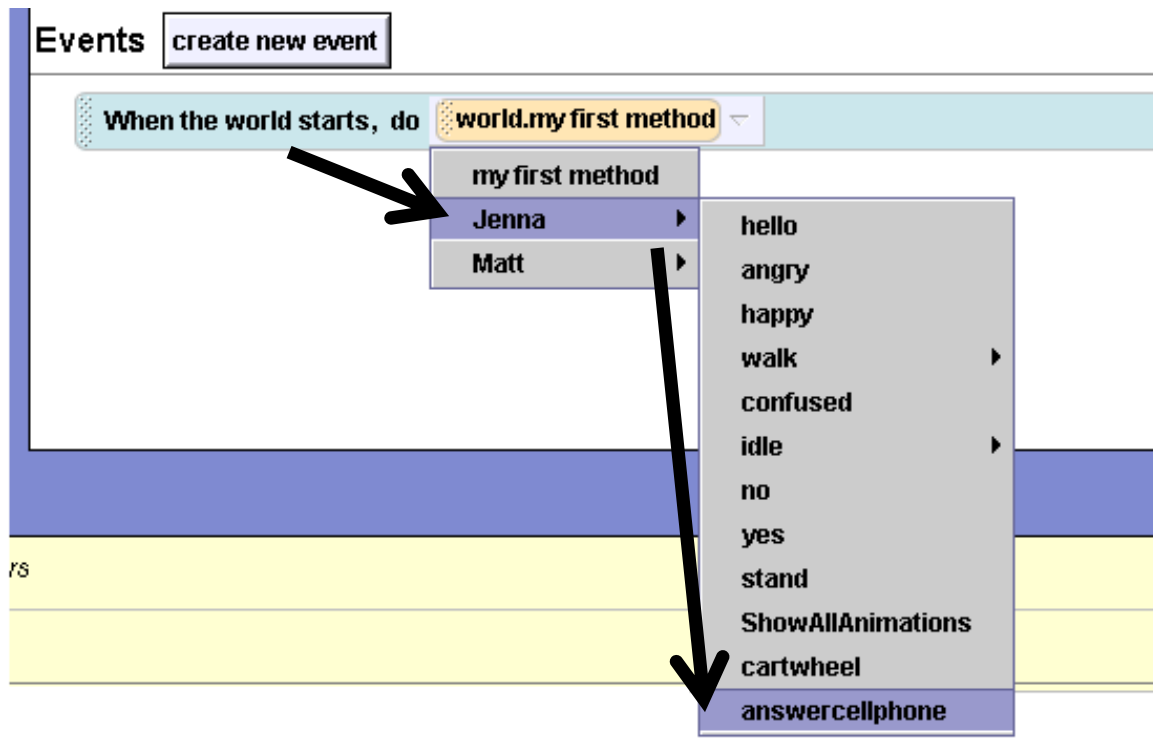
[-] Do together

Jenna.UpperBody.Chest.RightUpperArm ▾ roll left ▾ 0.18 revolutions ▾ more... ▾

Jenna.UpperBody.Chest.RightUpperArm.RightForearm ▾ roll left ▾ 0.5 revolutions ▾ more... ▾

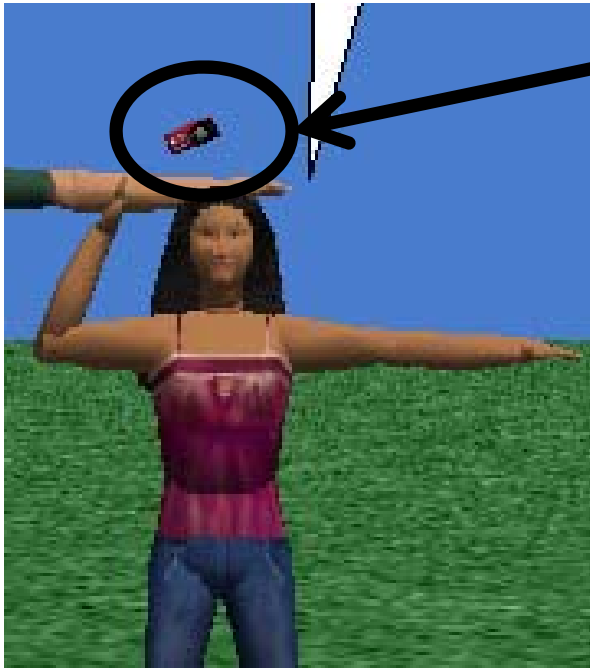
Jenna ▾ say hello ▾ more... ▾

Now you can try out your answercellphone method by going back to **when the world starts** and changing it to **Jenna.answercellphone**.



Press play to see what it looks like.

-Your cellphone method may still look a little bit wrong. If Jenna's phone looks like it is floating in the air, you should go back into quad view to reposition it so that it is right on her pocket. It may look right from the front, but when you view it from the side, you may see that the phone is actually several meters away from Jenna.



-If your phone is not close enough to Jenna's hand when she moves it up, for example if it is on her forearm, try using quad view to move the phone left and right to get the perfect position.

Method “Ride Horse”

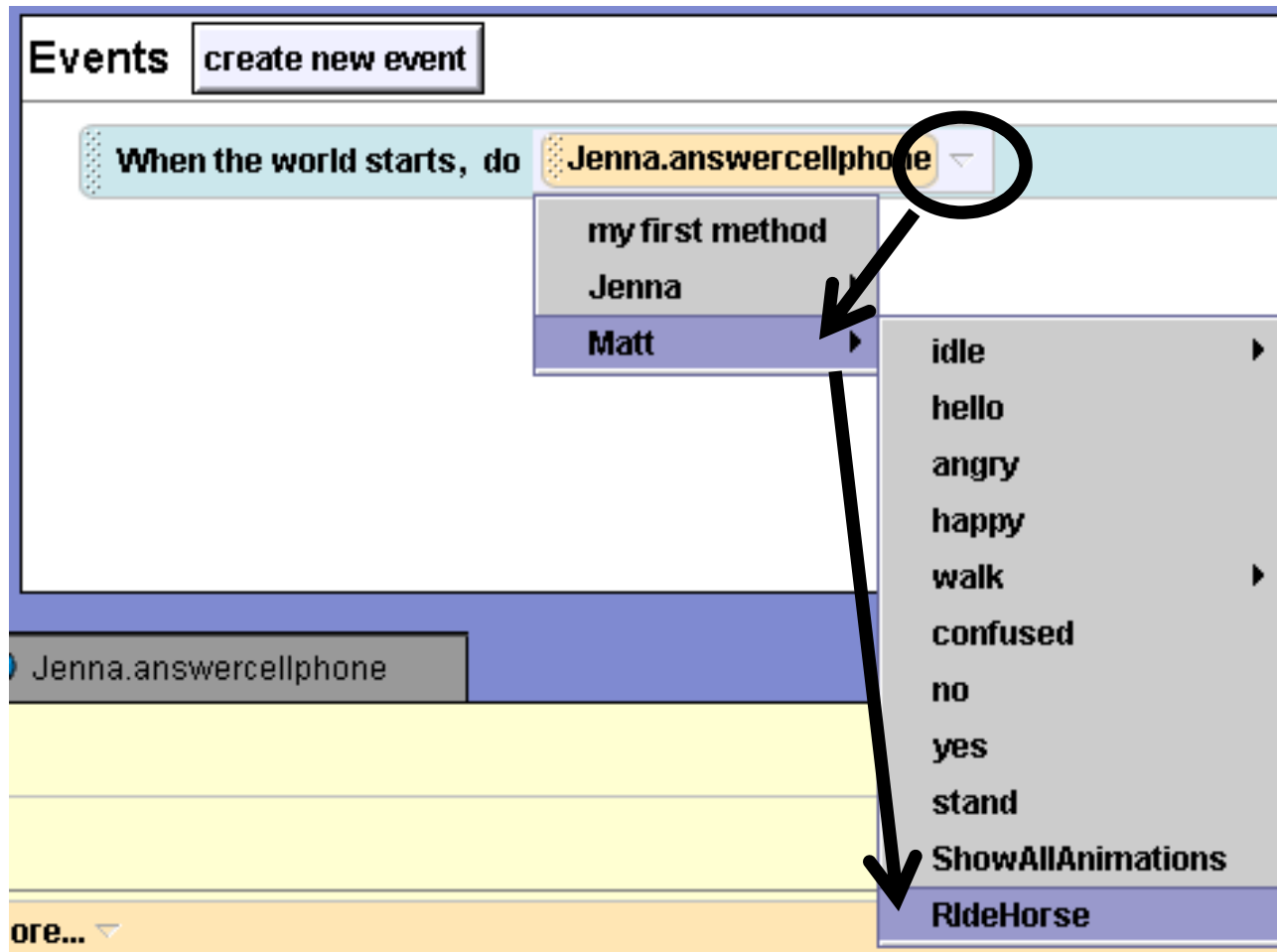
- Now lets create a method for the other character and the horse.
- Click on your other character in the object tree (we are doing it for Matt), and then click create new method. label the method “Ride Horse”.
- Lets break this down by steps
- First, we need the character to turn and face the horse



- Then we need the character to get on the horse
- Finally we need him to face forward again

Playing Your “Ride Horse” Method

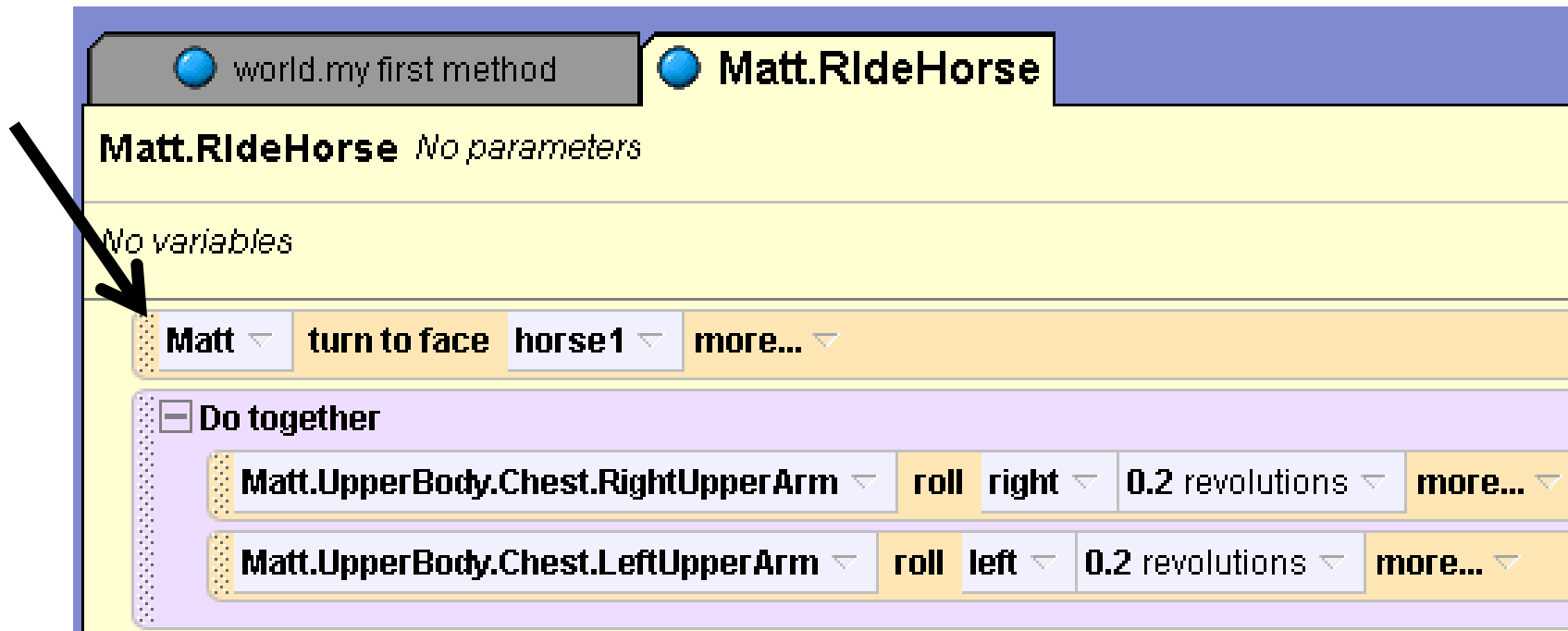
-In order to see your Ride Horse method when you play your world, you need to change your **when the world starts** event again. Change it from **answercellphone** to **Ride Horse**.



-Each time you add code to **Ride Horse** you should play your world to test it out.

Method “Ride Horse”

- For the first part we will simply write code to turn the character towards the horse and put his arms down.



Test this code by playing your world. If your characters arms do something other than moving down, you may have to try using a **turn** method instead of a **roll** method. This is because of the different sizes and shapes of the people-builder objects.

Method “Ride Horse”

- To code the character walking towards the horse, you simply need to move his limbs and keep playing it until you are happy with his movements. Add this code to the end of **Ride Horse**.



You may need to adjust these numbers according to the position of your character. If Matt does not go up high enough, make him move farther up. If he goes up too far, don't make him move quite as much.



If Matt looks like this, you may need to tell him to move up less, or move down more.



If Matt looks like this, you may need to tell him to move up more.

Method “Ride Horse”

- The final part will be to code our character to face the same direction as the horse.



- We can also tilt his shins back to look more realistic.

Do together

Matt.LowerBody.RightThigh.RightShin turn forward 0.1 revolutions more...

Matt.LowerBody.LeftThigh.LeftShin turn forward 0.1 revolutions more...

Add this code to the end of Ride Horse.

Finishing “Ride Horse”

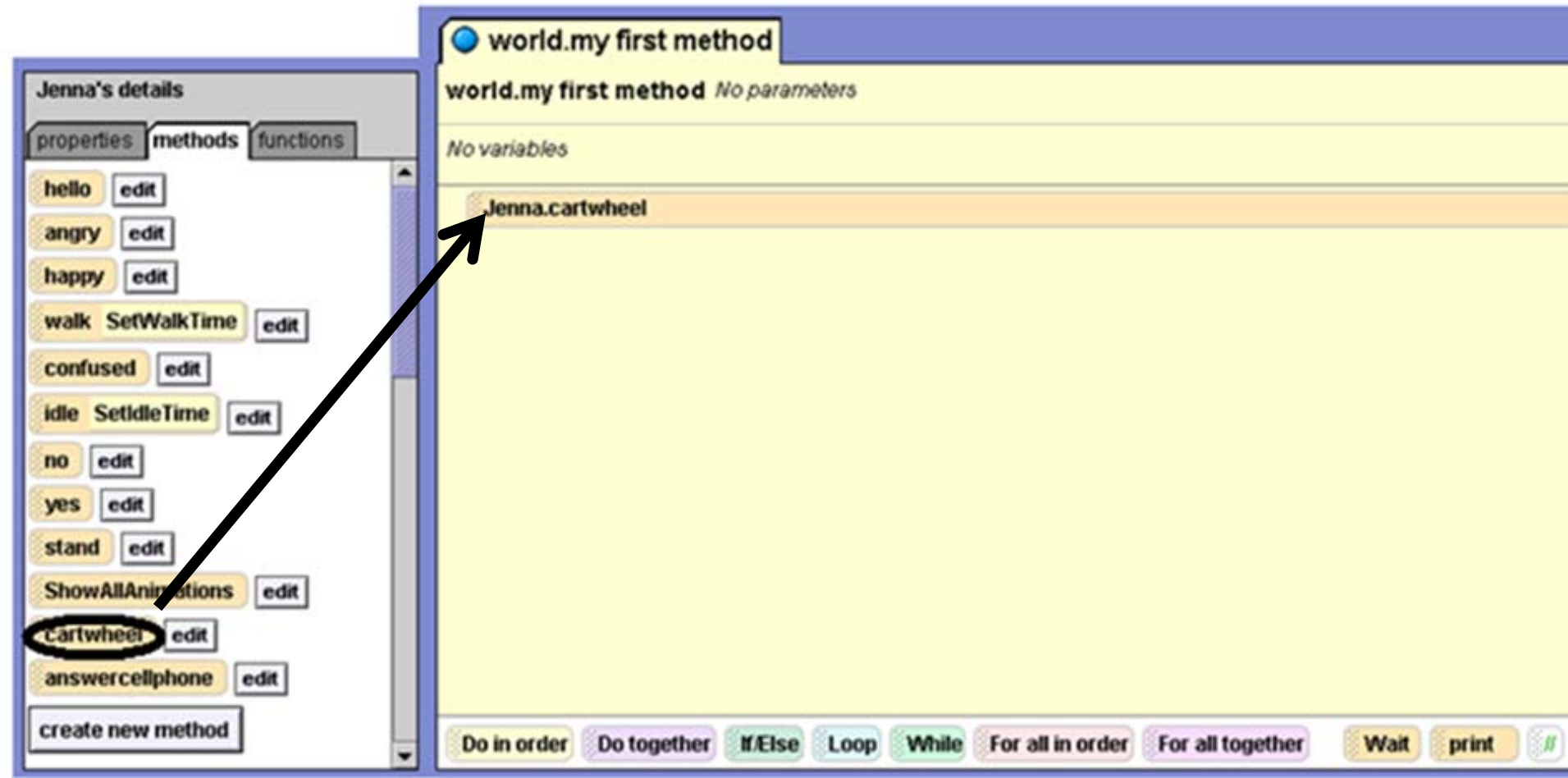


- Your final product should look something like this. Remember, your numbers may need to be different than the ones suggested here.
- Remember to keep playing your world as you code.
- If you code something wrong it is easy to see when you play the animation
- The “Undo” button is your friend!

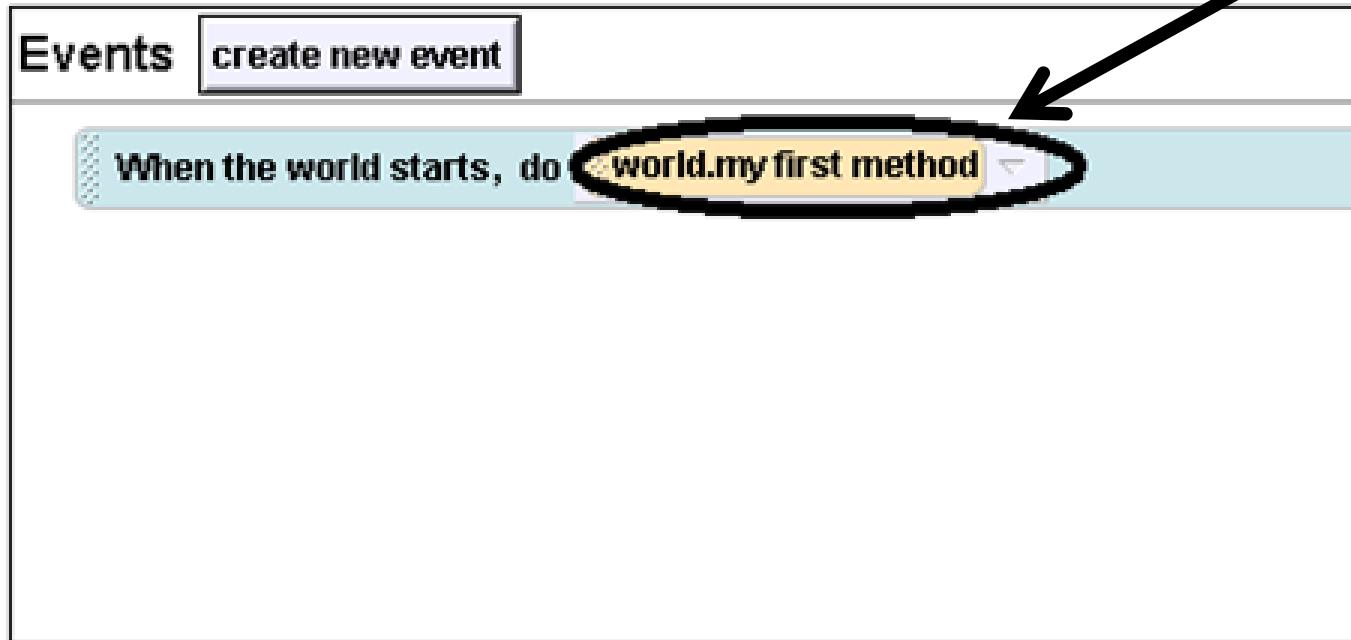
Now that you know how to write a method for a specific object, we are going to put these methods together to make a story. Click on the **world.my first method** tab on your screen. You may have code in there already from when you were testing out the he-builder/she-builder methods. You can keep it if you like, or erase it and add it back in later. These slides will start with a fresh, empty **my first method**.



This is where we will keep the parts of our story. First, we will use the cartwheel method that we taught to Jenna. Click on **Jenna** in the list of objects on the left side of the screen, and then the **methods** tab to find cartwheel again. Drag cartwheel into **my first method**.

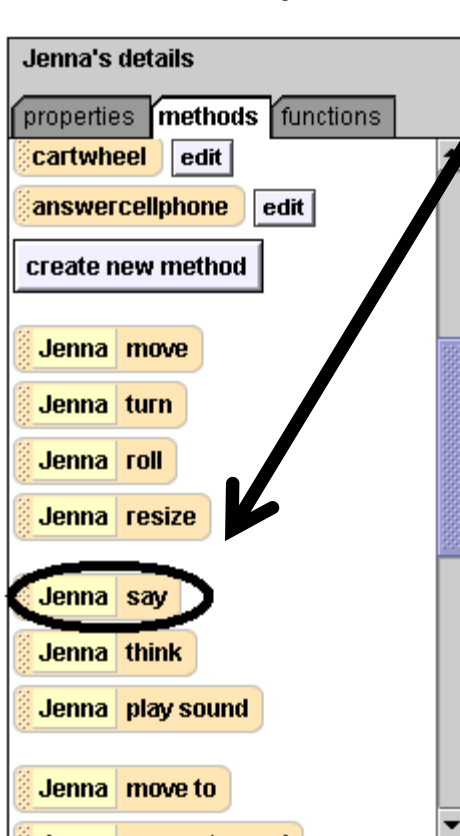


In the right hand corner of your screen find where it says **when the world starts do**, and change it to **world.my first method**. This way whenever you play your world it will play the story we will have in **my first method**.

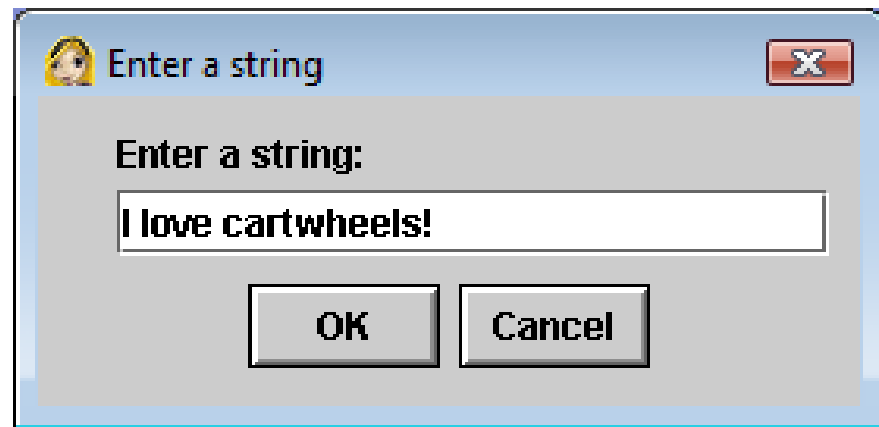


Now play your world to see what it looks like so far. It should just show one of your characters cartwheeling.

Now we want to make our character say something. Look at the methods tab and scroll down until you see **say**.



Now drag it into **my first method** under cartwheel. Click on **other...**, and you will have to type in something for your character to say. Type in something like "I love cartwheels!"



Play your world again to see the results.

Now we want to change it so that your character cartwheels twice in a row, to show just how much she loves cartwheels. To do this, we will use a **loop**. The loop button is located at the bottom of your method editor. Drag it into **my first method** above cartwheel. When you drop it select **2 times**.

The image shows a Scratch-style method editor for a character named 'world'. The method is titled 'world.my first method' and has 'No parameters' and 'No variables'. The method body contains three blocks: a 'Loop' block, a 'Jenna.cartwheel' block, and a 'Jenna' block with a 'say' block and a 'more...' block. The 'Loop' block is currently set to '2 times' and 'times'. A black arrow points from the 'Loop' button in the bottom toolbar to the 'Loop' block in the method body. The 'Loop' button in the bottom toolbar is circled in red.

world.my first method

world.my first method No parameters

No variables

Loop 2 times times show complicated version

(Do Nothing)

Jenna.cartwheel

Jenna say I love cartwheels! more...

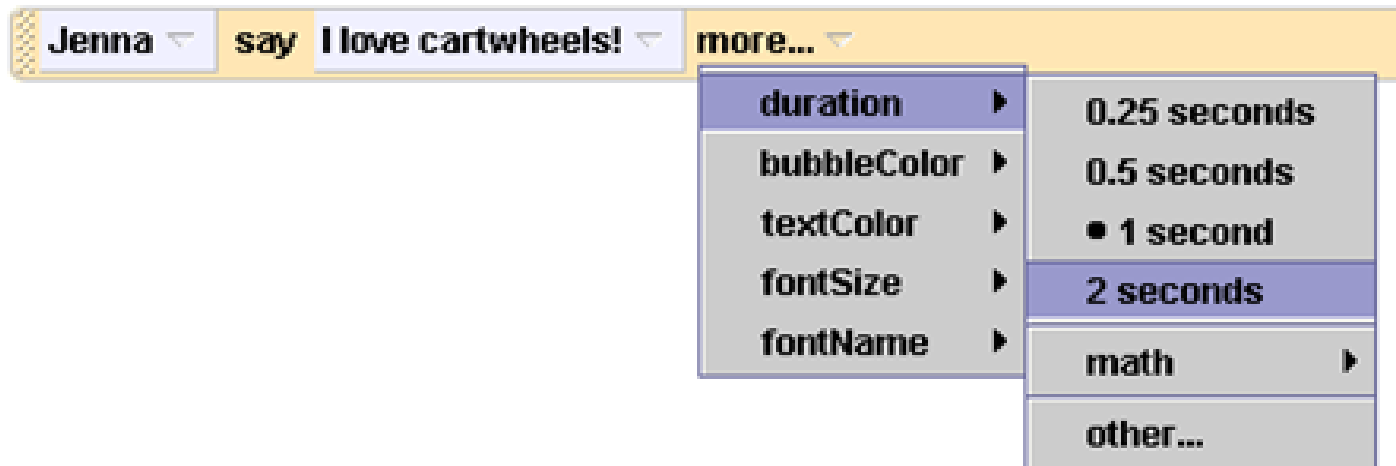
Do in order Do together If/Else Loop While For all in order For all together Wait print

Now drag and drop your cartwheel command inside the **Loop**. In order to grab the cartwheel command to drag it, you must click on the left-most part of the method. Your cartwheel will now run 2 times! Play your world to see what happens.

Grab your method here to drag it.

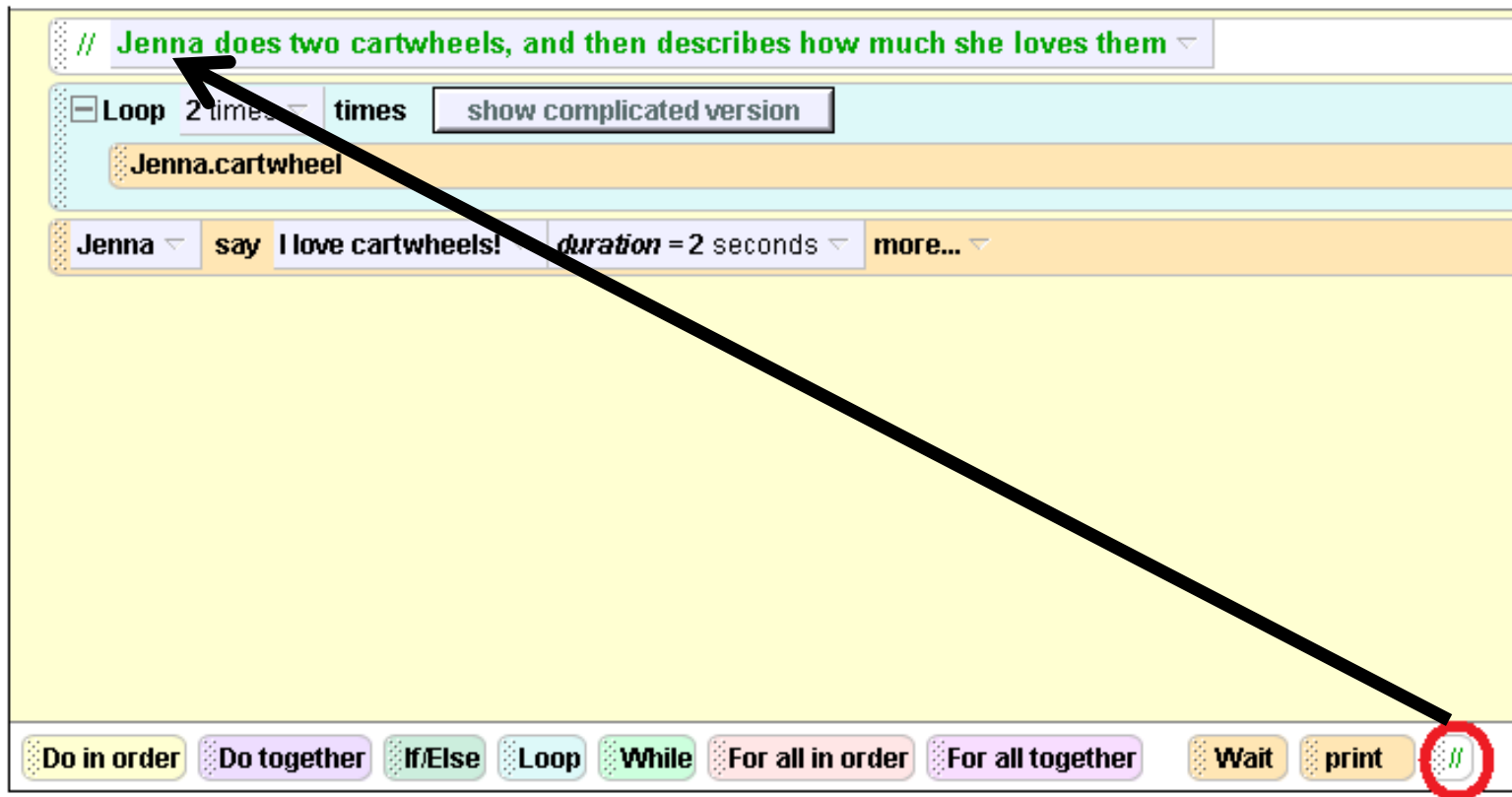
The image shows a Scratch code editor window. At the top, there is a tab labeled "world.my first method". Below the tab, the text "world.my first method No parameters" is displayed. Underneath, it says "No variables". The main code area contains a "Loop" block set to "2 times" with a "show complicated version" button. Inside the loop is a "Jenna.cartwheel" block. Below the loop is a "Jenna" block with a "say I love cartwheels! more..." block. At the bottom of the editor, there is a palette with various control blocks: "Do in order", "Do together", "If/Else", "Loop", "While", "For all in order", "For all together", "Wait", "print", and a comment block. A black arrow points from the text "Grab your method here to drag it." to the left-most part of the "Jenna.cartwheel" block.

You may feel that your character's speech bubble disappears too quickly. If this is true, there is a way to make it last longer. Click on **more...** at the end of the say command. Then click on **duration**. As you can see, it lasts 1 second. You can make this longer if you wish, maybe 2 or 3 seconds.



Play your world to test out different durations until you find one you like.

Now we are going to add **comments**. These are descriptions that you put in your code so that you, or anyone else that reads your code, can see what it does. The comment button is located at the bottom of the method editor. Drag it into the top of **my first method** and type a quick description of what your code does.



Now we will make your character answer her phone. Find the **answercellphone** method and drag it into **my first method**. Play your world again to see your changes.

The image shows a programming interface with two main panels. On the left is the 'Jenna's details' panel, which has tabs for 'properties', 'methods', and 'functions'. The 'methods' tab is selected, showing a list of methods: 'hello', 'angry', 'happy', 'walk', 'SetWalkTime', 'confused', 'idle', 'SetIdleTime', 'no', 'yes', 'stand', 'ShowAllAnimations', 'cartwheel', 'answercellphone', and 'create new method'. The 'answercellphone' method is circled in red. On the right is the 'world.my first method' script area. It has a title bar 'world.my first method' and a subtitle 'world.my first method No parameters'. Below this, it says 'No variables'. The script area contains a comment '// Jenna does two cartwheels, and then describes how much she loves them', followed by a 'Loop' block set to '2 times' with a 'show complicated version' button. Inside the loop is a '.Jenna.cartwheel' block. Below the loop is a 'Jenna say I love cartwheels! duration = 2 seconds more...' block. At the bottom of the script area is a 'Jenna.answercellphone' block. A black arrow points from the 'answercellphone' method in the left panel to the 'Jenna.answercellphone' block in the script area. At the bottom of the interface is a palette with various control blocks: 'Do in order', 'Do together', 'If/Else', 'Loop', 'While', 'For all in order', 'For all together', 'Wait', 'print', and a green flag icon.

Now we'll add one more method to the story. Click on Matt, your other character, in the list of objects, and look at his methods until you find the RideHorse method. Add it to **my first method**. Then add another comment above **answercellphone** that describes the rest of your code.

The screenshot shows the Scratch IDE interface. On the left, the 'Matt's details' sidebar is open, showing the 'methods' tab. The 'RideHorse' method is circled in the list. On the right, the 'world.my first method' script area is open, showing a script with a loop and a comment. The 'RideHorse' block is added to the script. The 'Loop' block is also visible in the script area. The 'RideHorse' block is circled in the script area. The 'Loop' block is also visible in the script area. The 'RideHorse' block is circled in the script area. The 'Loop' block is also visible in the script area.

Matt's details

properties methods functions

idle SetIdleTime edit

hello edit

angry edit

happy edit

walk SetWalkTime edit

confused edit

no edit

yes edit

stand edit

ShowAllAnimations edit

RideHorse edit

create new method

Matt move

world.my first method

world.my first method No parameters

No variables

// Jenna does two cartwheels, and then describes how much she loves them

Loop 2 times times show complicated version

Jenna.cartwheel

Jenna say I love cartwheels! duration = 2 seconds more...

// Jenna answers her phone and says hello, and then Matt gets on the horse.

Jenna.answercellphone

Matt.RideHorse

Do in order Do together If/Else Loop While For all in order For all together Wait print

Congratulations on creating a great story! Now we are ready to move on to Part 3 to learn new things about changing the camera view.



Alice

Learning to program: Part Three
Camera Control, Invisibility, and 3-D Text



Camera Control

- Now that we have a great story written for our characters it would be nice to be able to change camera views.
- This can simplify the story and sometimes add dramatic effect

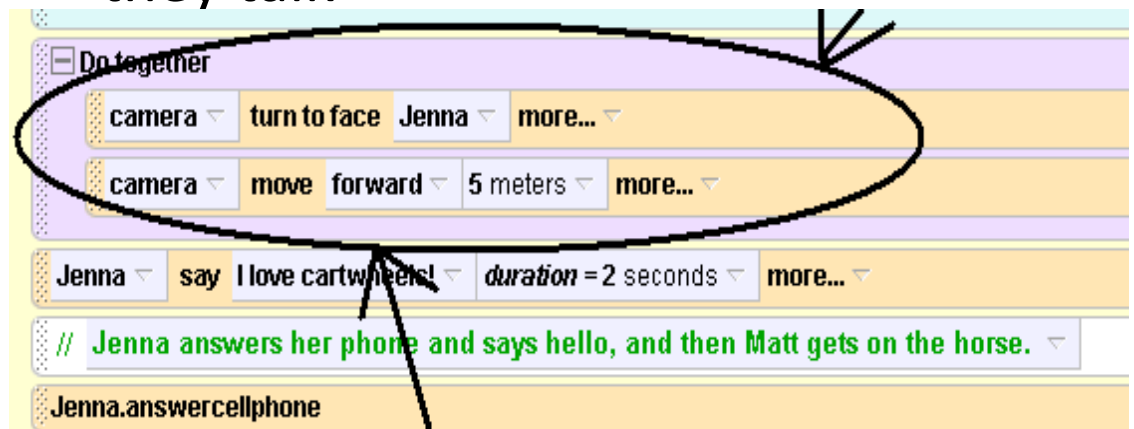
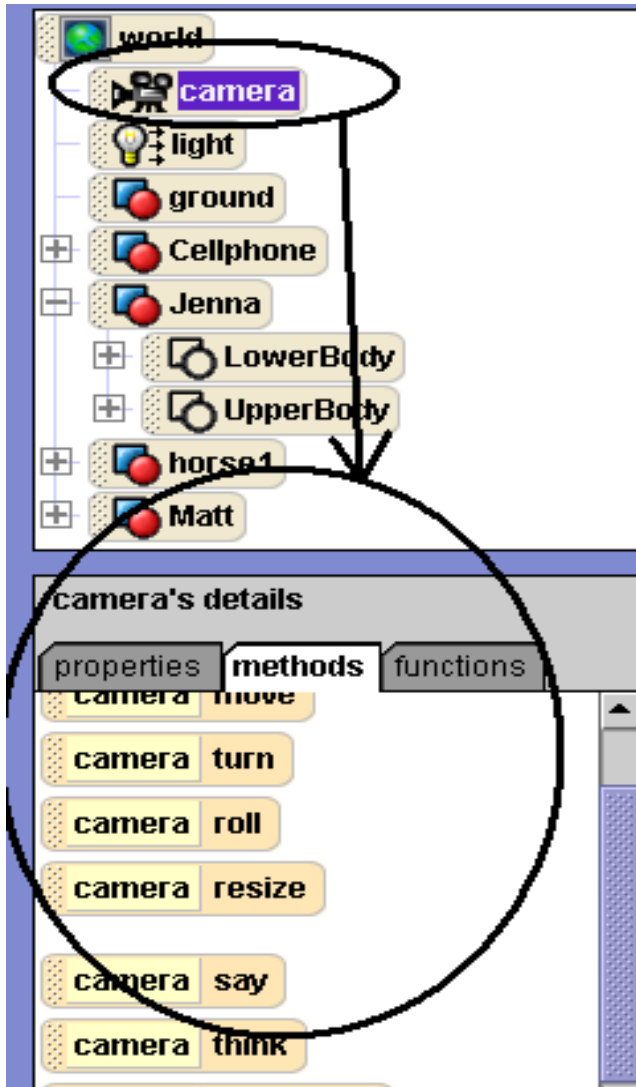
Camera Control



- Lets make the Camera Zoom in on our first character when she talks and then look at the horse while our second character gets on.

Camera Control

- Click on “camera” in your object tree.
- It should have a regular set of methods
- Insert code into your first method to have the camera face your character while they talk



Camera Control

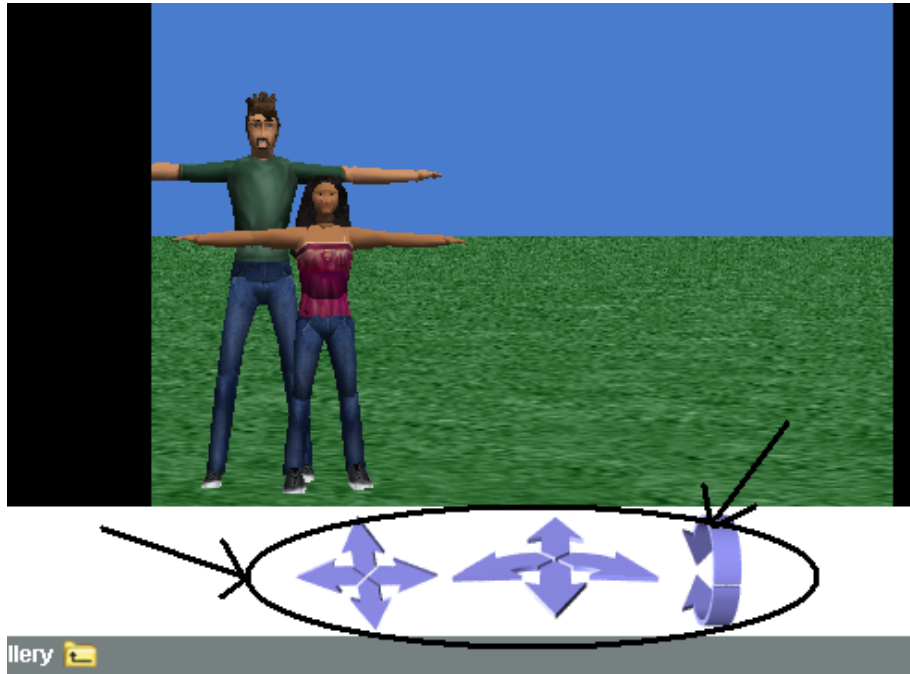


- When your second character gets on the horse, you can have the camera face him. Insert this code above **Matt.RideHorse**.



- Finally, have the camera face the horse and press the "N" key to have your horse say "neigh", when the camera is facing the horse. Play your world to test it and press the N key.

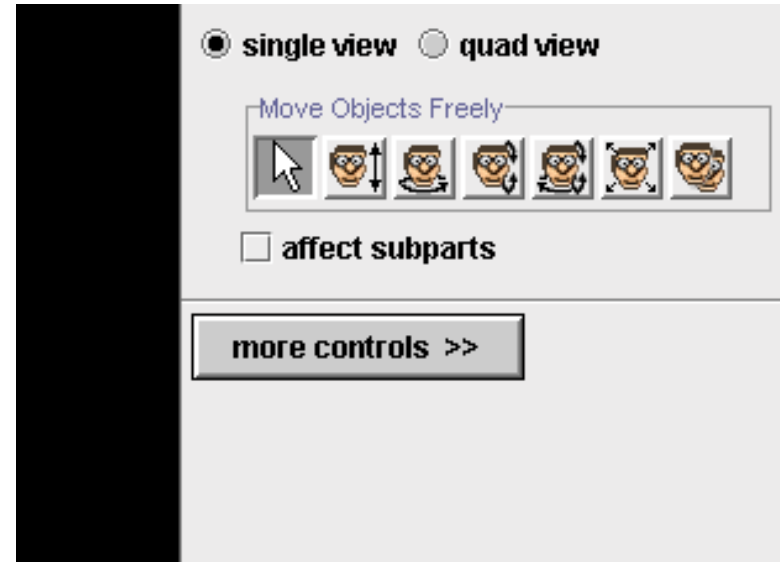
Dummy Cameras



- You can use the purple arrows on the bottom of your world to move around the camera
- You can drop a dummy camera wherever you find a view that needs a snapshot

Dummy Cameras

- Dummy cameras are used to hold a specific view of your world in place while you move the regular camera around
- Click the add objects button on your screen



- Look to the right for a button labeled “more controls”. Click it

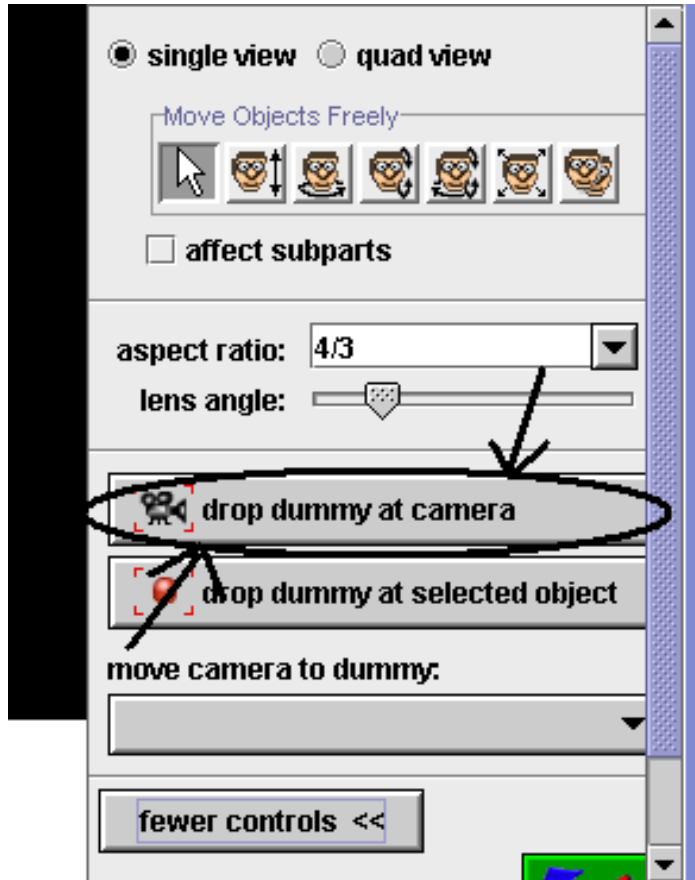
Dummy Camera

- We did this earlier in part one by saving the original camera position, but we will now add more camera positions. You should always save your original camera position before moving your camera.
- Use the purple arrows to move the screen to a new position on your camera that looks closely at the horse.



-We will show you how to drop a dummy camera at this location and name it **Horse Position**.

Dummy Cameras



- Now click on “drop dummy at camera”. Just click it once!
- This will literally drop a virtual camera where your camera currently is.

Dummy Camera

- Go over to your object tree
- There should be a folder called “Dummy Objects”.
- Underneath it will be a list of each dummy you have dropped, in the order that they are dropped

We renamed this earlier.

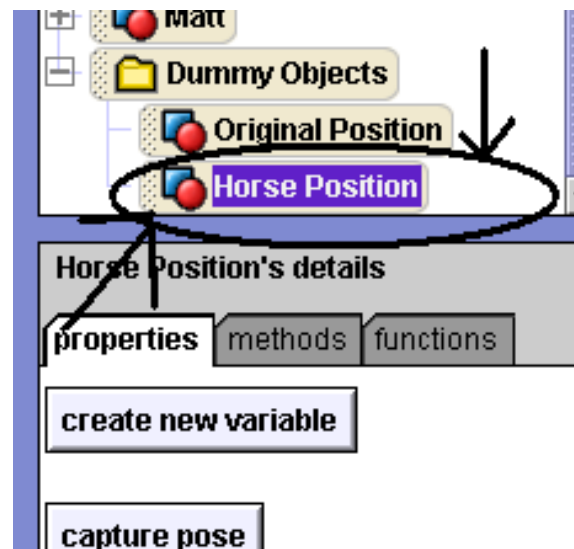
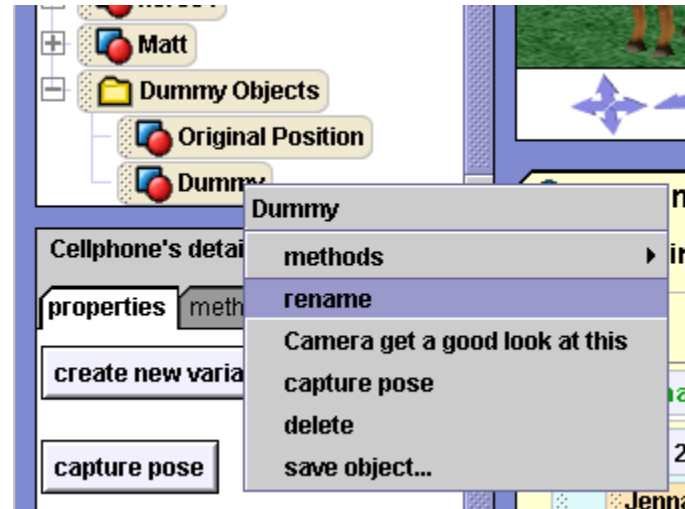


- Note that the first Dummy object is named “Original Position”. You added this back in part one.

This is our new dummy camera position.

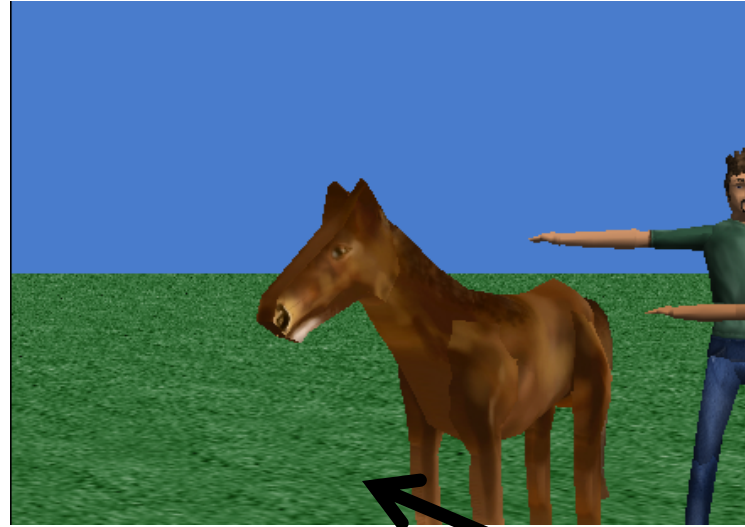
Dummy Camera

- Unfortunately all of your dummy cameras are labeled “Dummy”
- You can fix this by right clicking on “Dummy” and selecting “rename”.
- You can rename your Dummy whatever you would like.
- In this case it is named “Horse Position”.



Dummy Camera

- You can move the camera wherever you would like. Let's move it back to the original camera view.
- Now right click on "camera" in your object tree and select methods.
- Click "set point of view to" Dummy Objects/Original Position. See the next slide for a picture of the selection process.



Original position

Horse position



Resetting Your Camera View

The screenshot illustrates the process of resetting the camera view in a software application. The interface includes a left sidebar with a 'camera' object selected, a central workspace with a 3D scene, and a right sidebar with a 'method' list.

Step 1: The 'methods' menu for the 'camera' object is open. The 'camera set point of view to' method is highlighted.

Step 2: The 'camera set point of view to' method's submenu is open. The 'Dummy Objects' option is highlighted.

Step 3: The 'original position' and 'horse position' options are highlighted in the 'Dummy Objects' submenu.

Method List (Right Sidebar):

- camera move
- camera turn
- camera roll
- camera resize
- camera say
- camera think
- camera play sound
- camera get a good look at
- camera move to
- camera move toward
- camera move away from
- camera orient to
- camera turn to face
- camera point at
- camera set point of view to
- camera set pose
- camera stand up
- camera set color to
- camera set opacity to
- camera set vehicle to
- camera set skin texture to
- camera set fillingStyle to

Method Details (Right Sidebar):

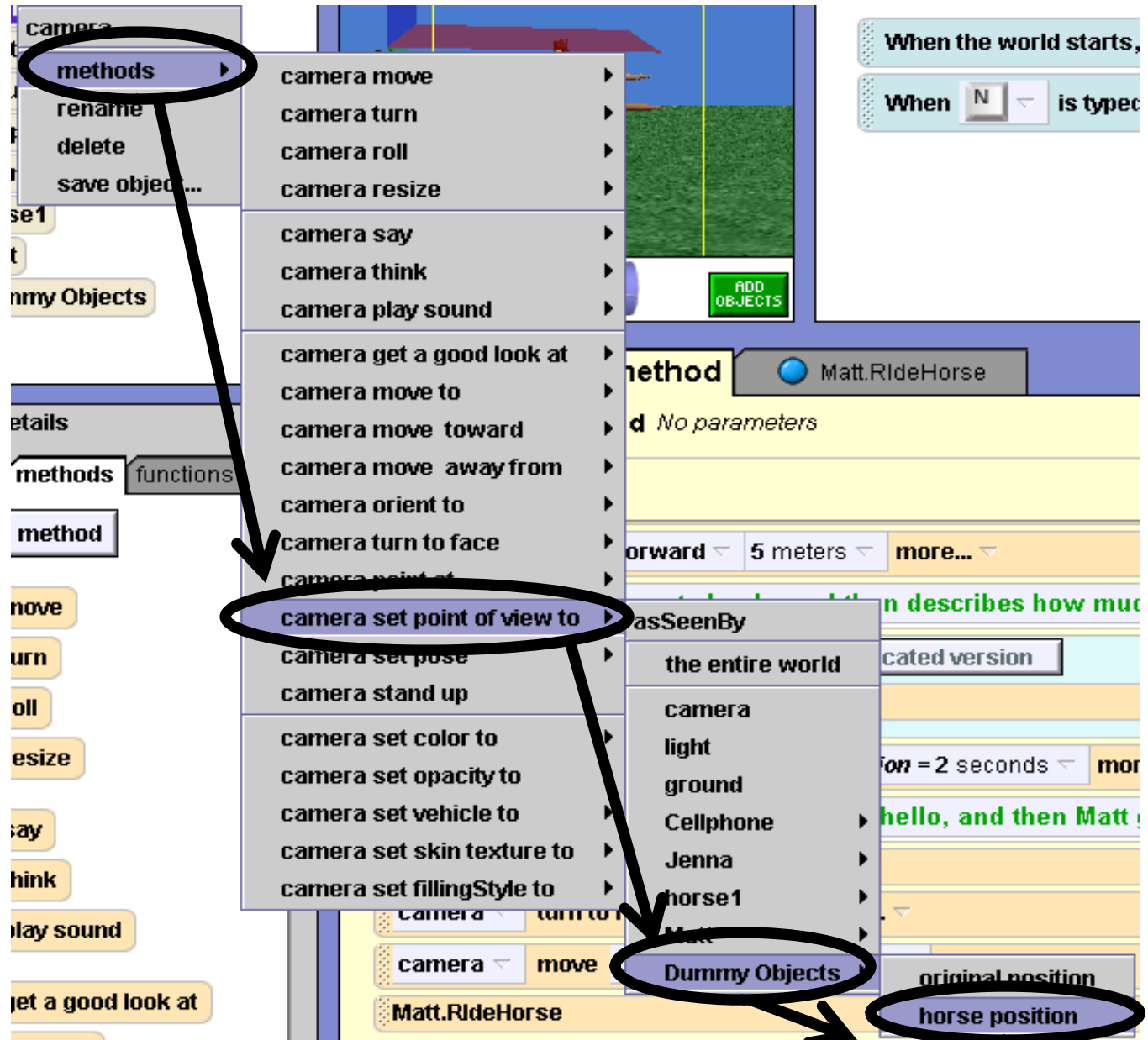
- Method: Matt.RideHorse
- Parameters: No parameters
- Forward: 5 meters
- More...: more...

Method List (Bottom):

- camera move
- camera turn
- camera roll
- camera resize
- camera say
- camera think
- camera play sound
- camera get a good look at

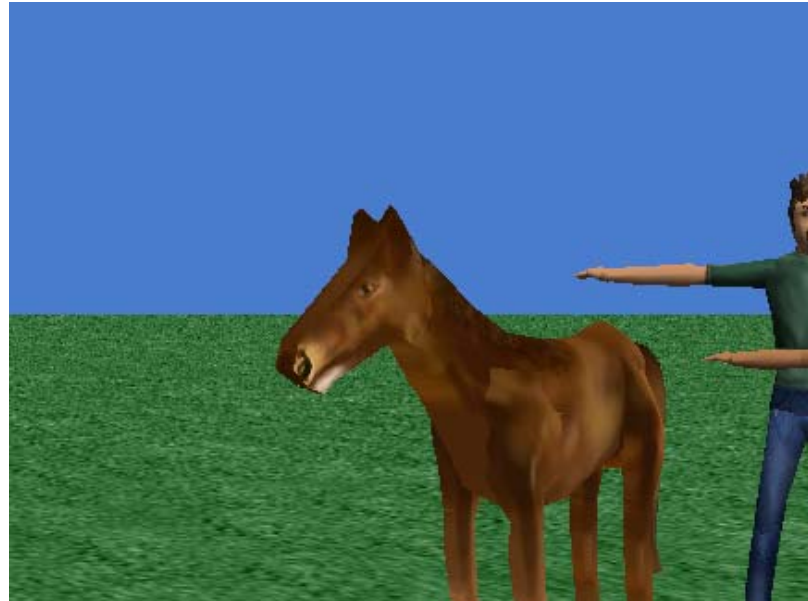
Resetting Your Camera View

-Now try using this same process to set your camera back to Horse View.



Dummy Camera

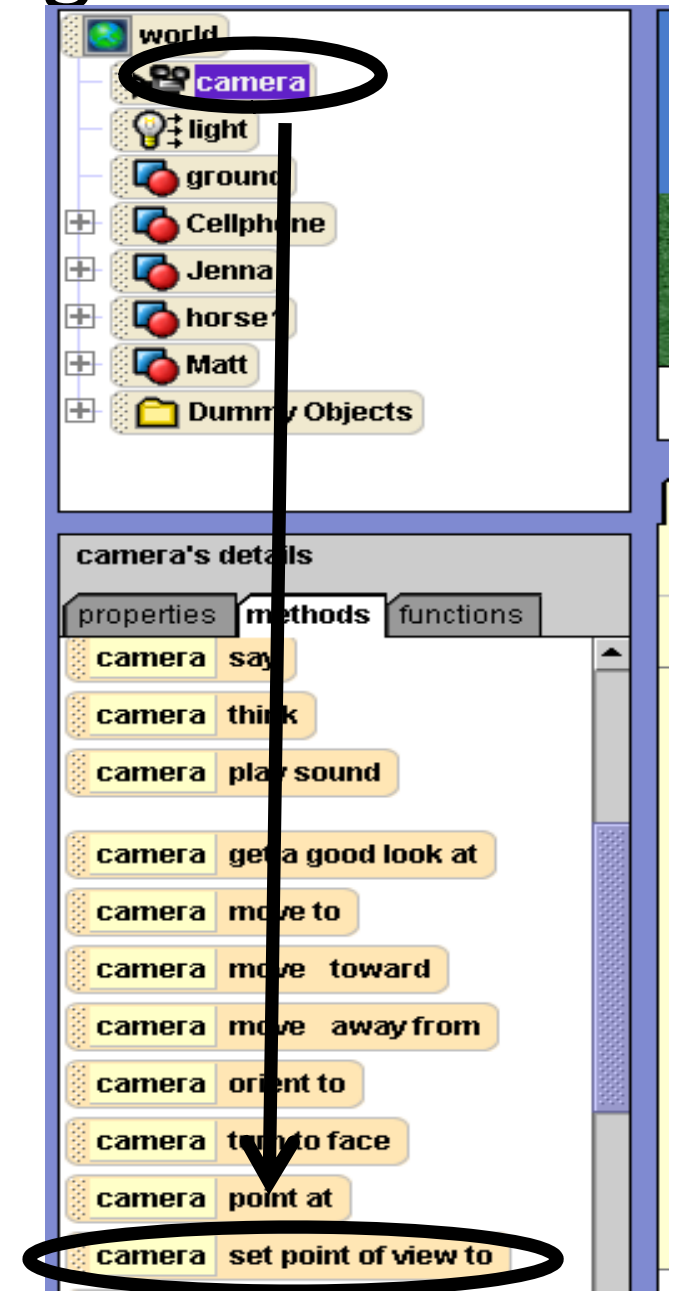
- Once you select “Horse Position” your screen should return to the position that it was at when you first dropped the Dummy Camera, for horse close up.



- You can use a Dummy Camera anywhere in the world that you would like.
- Changing the Camera View between Dummy Cameras, by right clicking and choosing “Methods” is useful for positioning things in your world, before it starts.

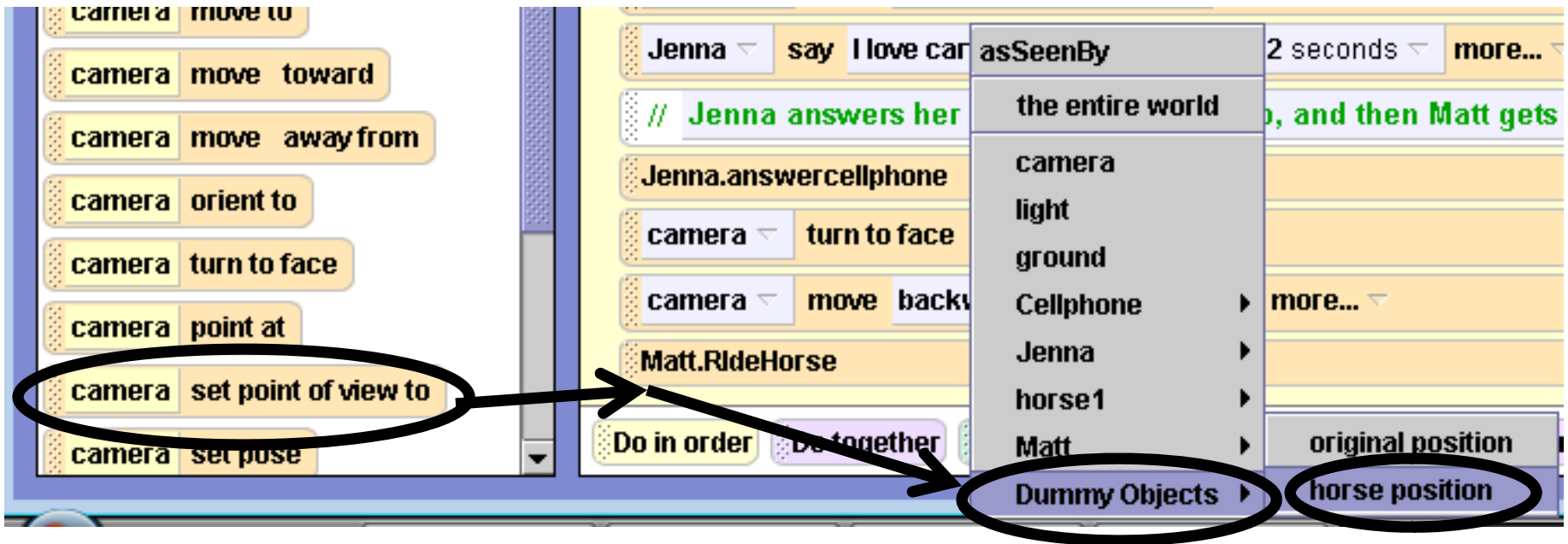
Dummy Cameras During Animation

- If you want the camera view to change during your animation you can drag it into your code, from camera methods.
- Click on camera in the object tree, and then find **set point of view to** in your methods tab.



Dummy Cameras During Animation

-Drag **set point of view to** to the very end of **my first method** right under **Matt.RideHorse** and drop it there. Set it to **Horse Position** so that the camera closes up on the horse at the end of your animation.



Play your world to test it out.

Moving On

-Now we will show you two new things:

-How to make objects all, or partly invisible.

-How to insert titles with text objects in your world.

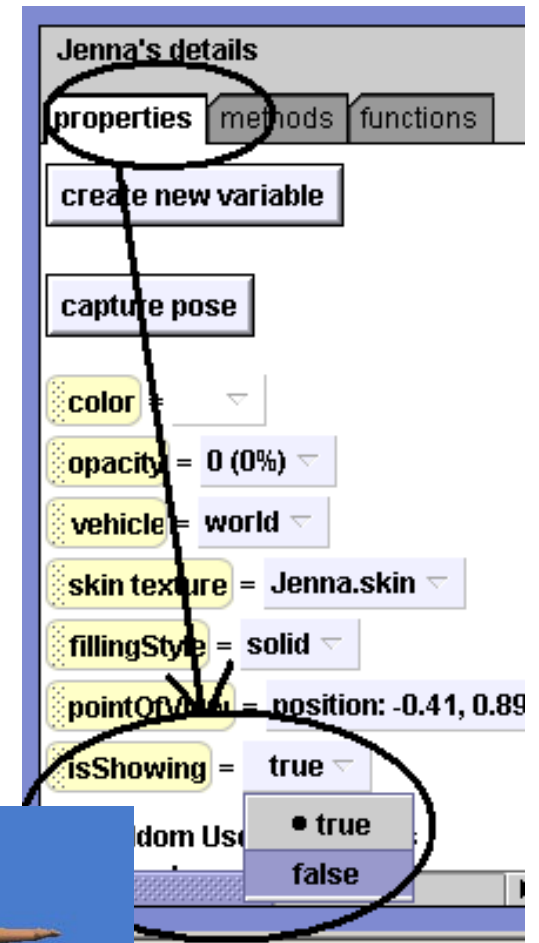
Making Objects Invisible:

IsShowing and Opacity

- Is showing and opacity both change your objects to make them more or less visible.
- Is showing has two settings. Your object is either visible, or its invisible.
- Opacity works by percentages. You can make something 10% visible, 20% visible, all the way up to 100% visible.

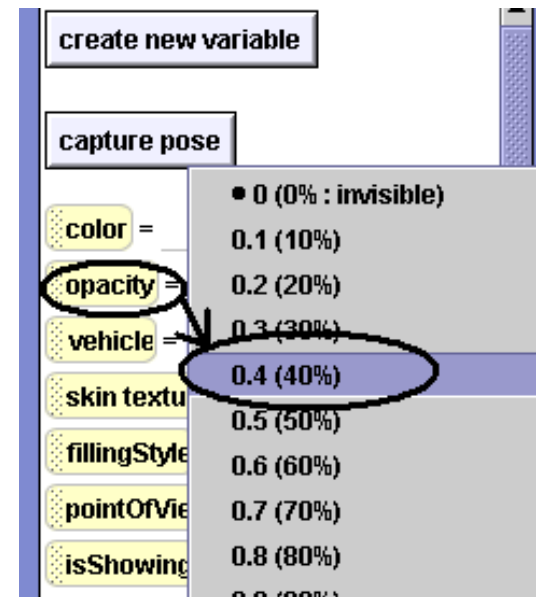
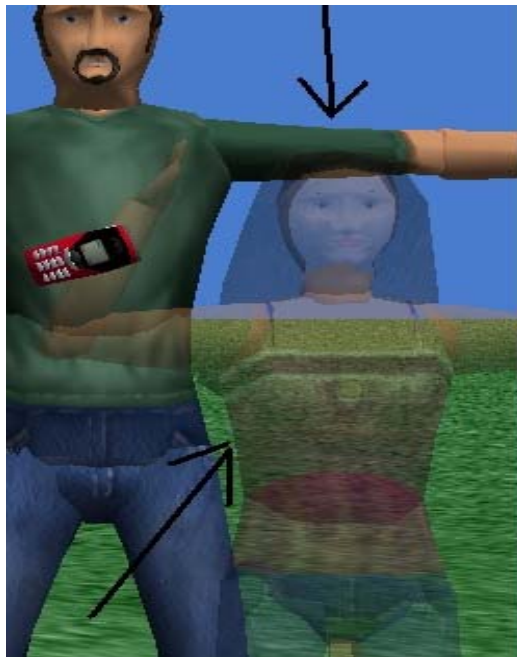
Invisibility :Is Showing

- Select the character that you would like to make invisible.
- Under the properties tab there should be a button called “Is showing”.
- If you select “false” your character should become invisible.
- Then make your character visible again by selecting “true”.



Making a Ghost:Opacity

- Opacity works in a very similar way to “Is showing”, except you can set an object to an in between stage.



- Try clicking on Opacity and selecting 40%
- Note that your character is now see through
- This works really well for ghost worlds!

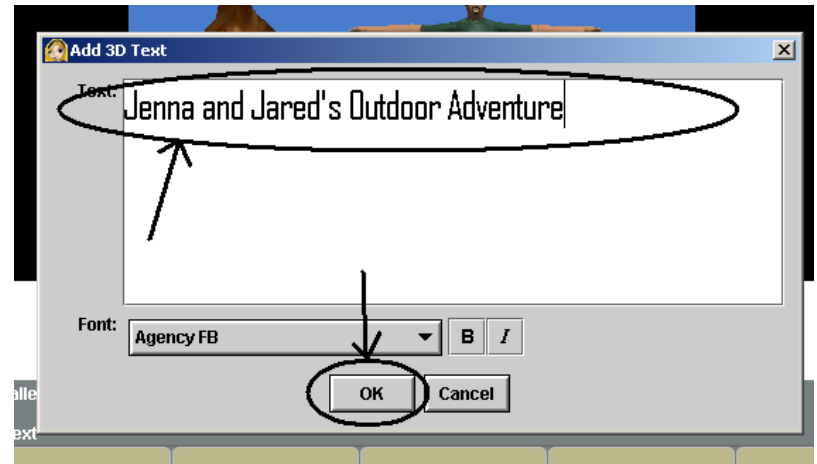
Adding Titles:3D Text

- Go into “Add Objects”
- Click on the “Create 3D text” Object



3D Text Continued

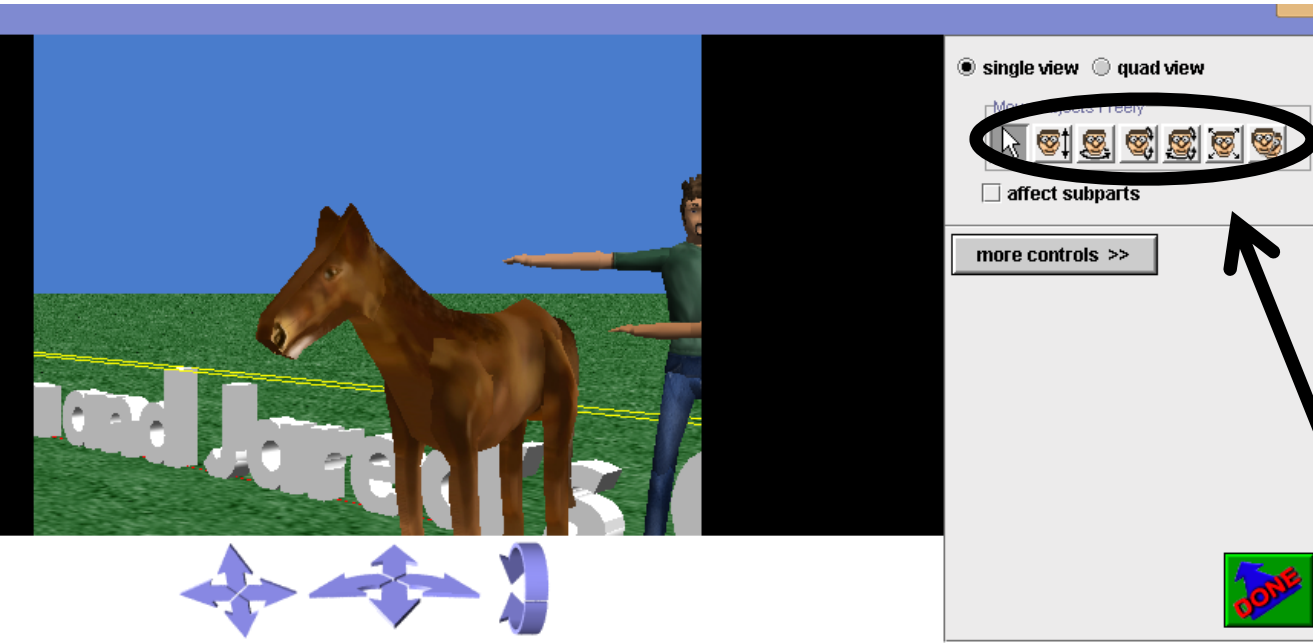
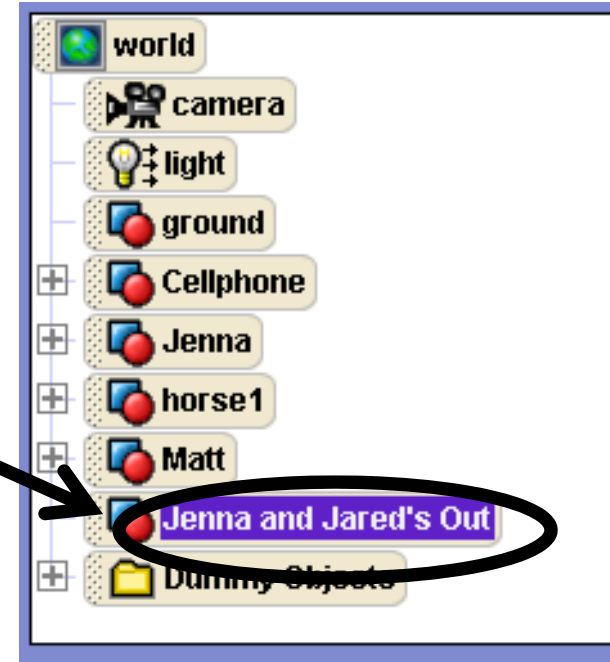
- Once you click on the 3D text object, a box should pop up to type text into.
- Type a message
- Select ok



- Your text should appear in your world like this

3D Text Continued

-Your text object will appear in your object tree just like any other object.



-You can also use the object moving buttons on it like any other object. Use them to position it in the sky.

3D Text



- This is how your text should look when you are done.
- You can set the “is showing” to false after a few seconds
- 3D text works really well for giving instructions at the beginning of a game or interactive story.

Finishing up



- Now you know the basics of creating a world
- In the next part, we will teach you some more details of Alice that can make your worlds really cool.

Problems

- Now its your turn, try completing these things in your world
 - Make an event so that one of your characters becomes invisible when you press **i**, and then another event that makes it visible again when you press **s**.
 - Make an event so that the Horse's opacity turns to 40% when you press **g**, for “ghost”.
 - Drop a Dummy Camera behind your characters and swap between that and Horse Position.

Alice

Learning to program: Part Four
Creating Sounds, Making Billboards, Fun with 3-D Text,
New Events, and Rotating Objects



www.cs.duke.edu/csed/alice/aliceInSchools

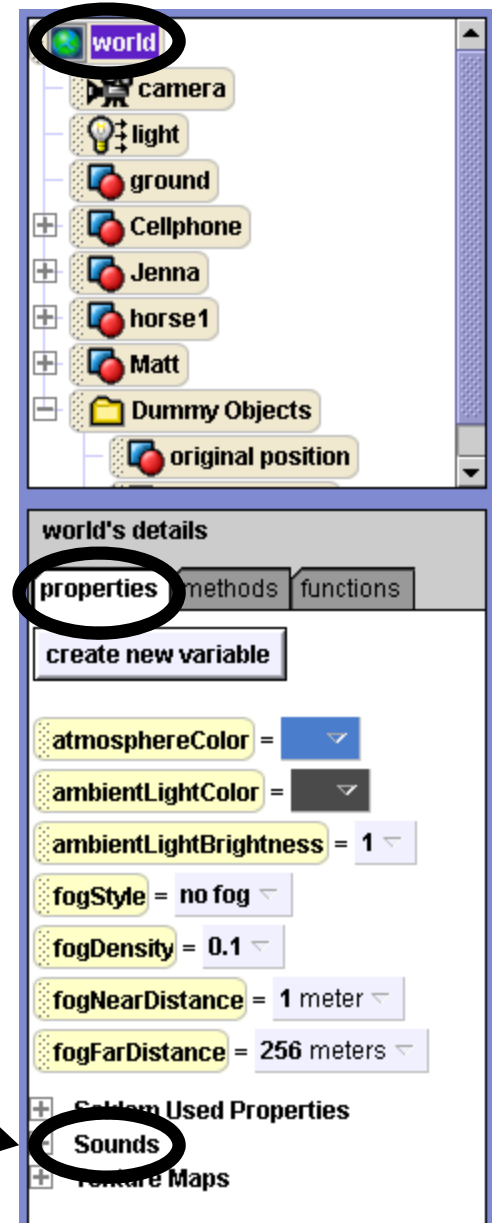
Creating a Sound in Alice

-We are now going to make your character actually speak.

-Click on **world** in the object tree.

-Go to the **properties** tab.

-Click on the plus sign next to **Sounds**.

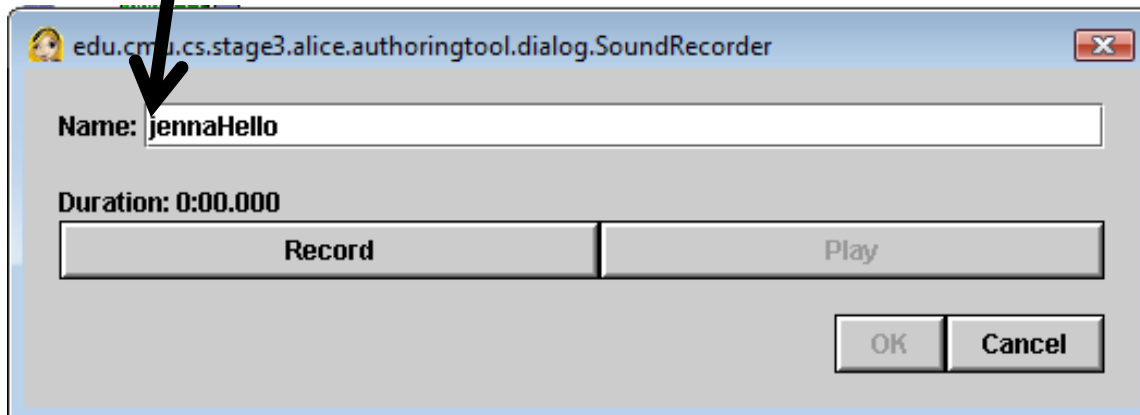
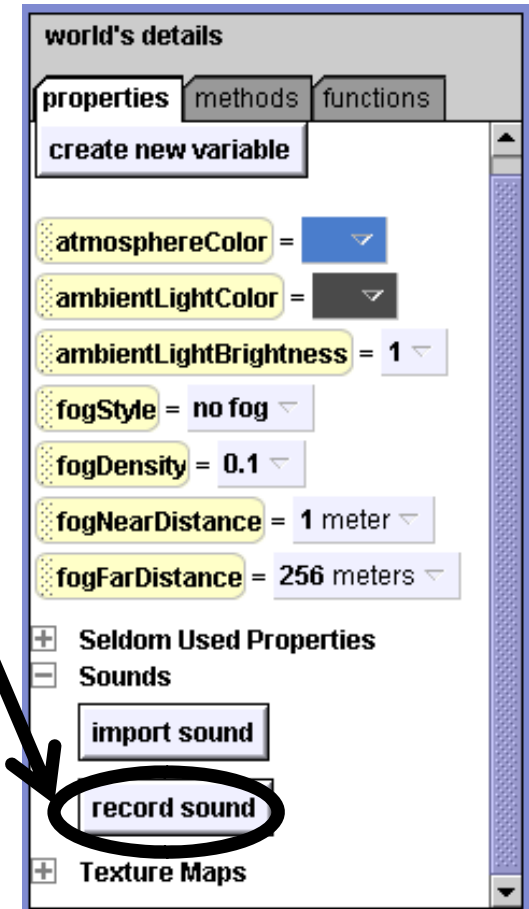


Naming the Sound

-Click on the **record sound** button.

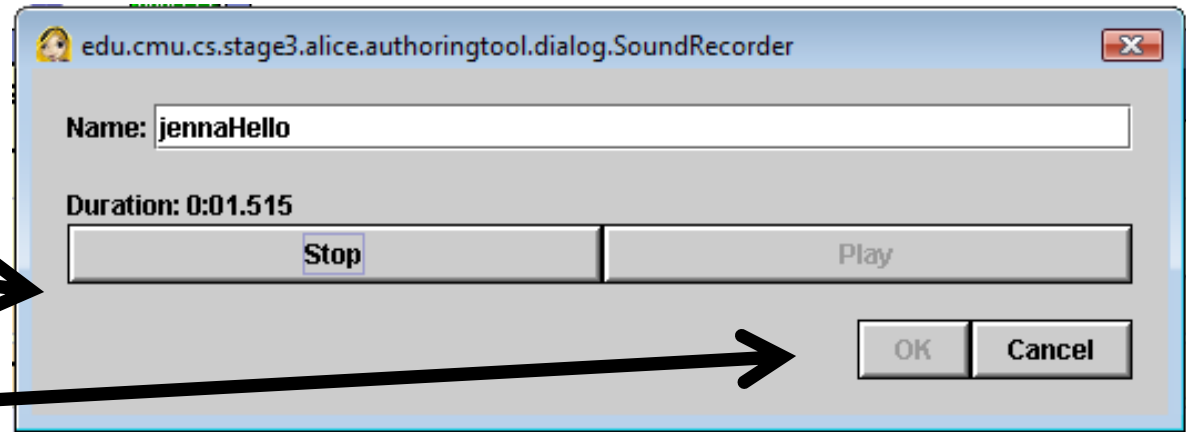
*(Make sure the volume on your computer is on)

-Type in a name for the sound file.
Call it **jennaHello**. We are going to make Jenna say “Hello?” when she answers her phone.



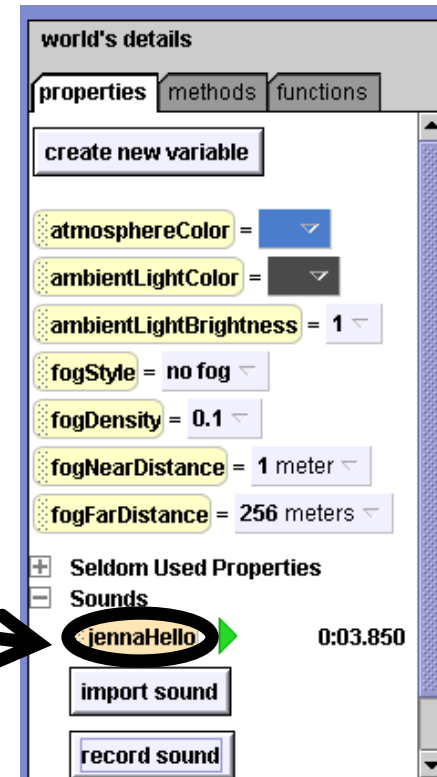
Recording the Sound

-Click **Record**, say “Hello?”, and then click **Stop** when you are done.



Click play to hear it. If you don't like it, record again and click ok.

-Your sound will appear on the **properties** pane under **Sounds**.



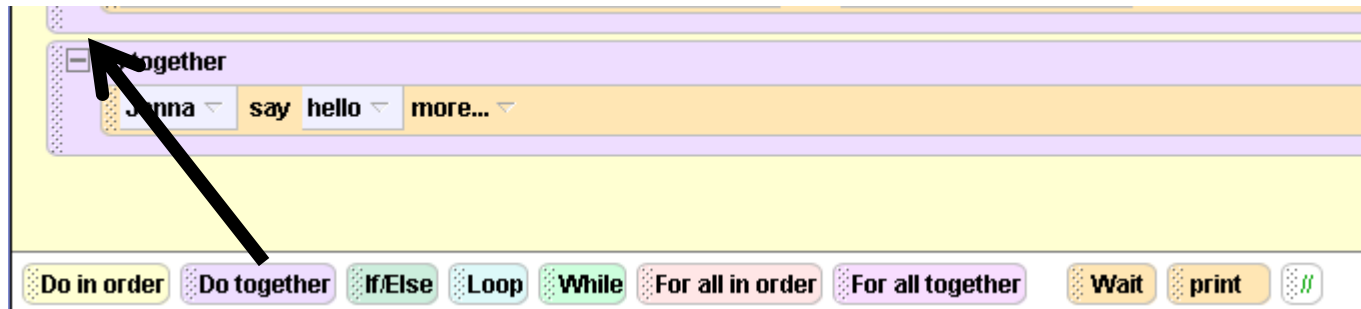
Editing answercellphone

-Click on **Jenna** in the object tree.

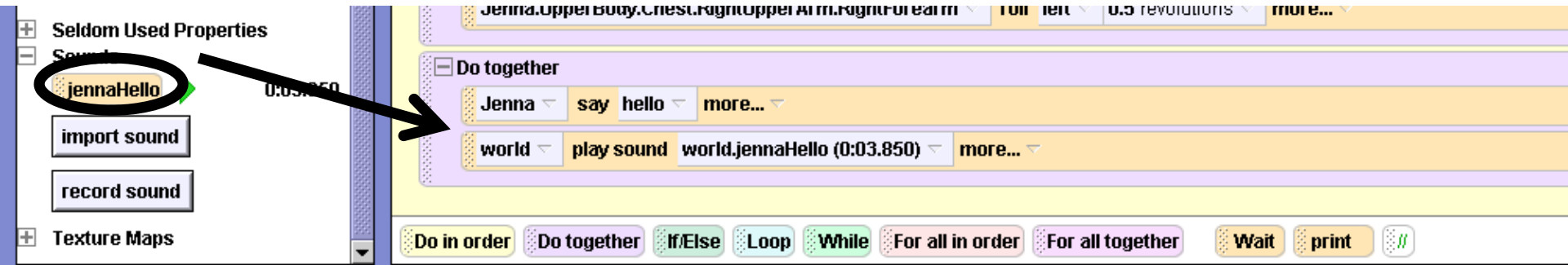
-Click on **methods**, and find **answercellphone**. Click on **edit** Next to that to see the code for **answercellphone**.

The image shows a software interface with two main panels. The top panel, titled 'Jenna's details', contains a list of objects in the 'object tree' on the left. The objects are: world, camera, light, ground, Cellphone, Jenna, horse1, Matt, Dummy Objects, and original position. An arrow points from the text 'Click on Jenna in the object tree.' to the 'Jenna' object. The bottom panel, titled 'Jenna's details', has three tabs: 'properties', 'methods', and 'functions'. The 'methods' tab is selected, showing a list of methods: hello, angry, happy, walk, SetWalkTime, edit, confused, edit, idle, SetIdleTime, edit, no, edit, yes, edit, stand, edit, ShowAllAnimations, edit, cartoon, edit, answercellphone, edit, and create new method. An arrow points from the text 'Click on methods, and find answercellphone. Click on edit Next to that to see the code for answercellphone.' to the 'answercellphone' method and its 'edit' button. Below the 'Jenna's details' panel, there is a small window showing the code for the 'Jenna.answercellphone' method. The window has a title bar with 'world.my first method' and 'Jenna.answercellphone'. The code area shows 'Jenna.answercellphone No parameters'.

Adding the Sound to your Code



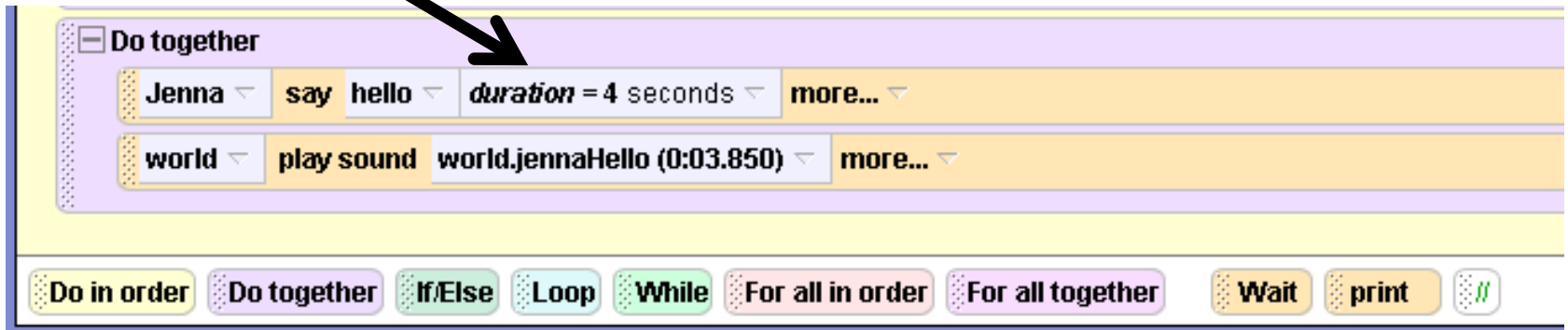
Drag a **Do together** into your code above your **Jenna say** and then drag the say method into it.



Find your sound again in world properties, and then drag it into the Do together with your say method.

Changing Duration

Set the duration of your **say** method so it matches approximately the length of your sound file.



Now your character can speak!
Play and try out the sound.

“As Seen By”



- “As Seen By” is a function that allows you to rotate an object around another object.
- We will make Jenna Circle around the horse.

As Seen By (Continued)

-First, drag a **camera set point of view to** command to the end of **my first method** and set it to **original position**, so you can see all your characters again at the end of your animation.

The image shows a screenshot of an animation software interface. On the left, a panel titled "camera's details" has three tabs: "properties", "methods", and "functions". The "methods" tab is selected, showing a list of camera commands. A black oval highlights the "camera set point of view to" command. A black arrow points from this command to the script area on the right. The script area is titled "world.my first method" and contains a sequence of commands. The last command in the script is "camera set point of view to original position", which is highlighted by the arrow. Other commands in the script include "Jenna.cartwheel", "Jenna say I love cartwheels! duration = 2 seconds", "Jenna answers her phone and says hello, and then Ma", "Jenna.answercellphone", "camera turn to face horse1 more...", "camera move backward 1.5 meters more...", and "Matt.RideHorse".

camera's details

properties methods functions

camera move toward

camera move away from

camera orient to

camera turn to face

camera point at

camera set point of view to

camera set pose

camera stand up

camera move at speed

camera turn at speed

camera roll at speed

world.my first method No parameters

No variables

Loop 2 times times show complicated version

Jenna.cartwheel

Jenna say I love cartwheels! duration = 2 seconds n

// Jenna answers her phone and says hello, and then Ma

Jenna.answercellphone

camera turn to face horse1 more...

camera move backward 1.5 meters more...

Matt.RideHorse

camera set point of view to horse position more...

camera set point of view to original position more...

As Seen By (Continued)

Now, click on **Jenna** in the object tree and find her **turn** method. Drag and drop it at the end of **my first method**. Tell her to turn **right 1 revolution**.

The screenshot shows a programming environment with two main panels. On the left, the 'Jenna's details' panel has tabs for 'properties', 'methods', and 'functions'. Under the 'methods' tab, a list of methods for the 'Jenna' object is shown: 'move', 'turn', 'roll', 'resize', 'say', 'think', 'play sound', 'move to', 'move toward', 'move away from', and 'orient to'. The 'turn' method is circled in black. On the right, the 'world.my first method' script area is shown. It contains a 'Loop' block set to '2 times' with a 'show complicated version' button. Below the loop is a 'Jenna.cartwheel' block. Further down is a 'Jenna.say' block with the text 'I love cartwheels!' and a 'duration' of '2 seconds'. Below that is a green flag block with the text '// Jenna answers her phone and says hello, and then Matt gets c'. Below the flag block is a 'Jenna.answercellphone' block. Below that is a 'camera.turn to face' block with 'horse1' as the target. Below that is a 'camera.move backward' block. Below that is a 'Matt.RideHorse' block. Below that is a 'camera.set direction' block with 'left' as the direction. Below that is a 'camera.set' block with 'right' as the direction. The 'right' option is circled in black. Below the 'camera.set' block is a 'Do in order' block. Below that is a 'Do to' block. Below that is a 'forward' block. Below that is a 'backward' block. The 'turn' method is being dragged from the 'Jenna's details' panel to the end of the script area. The 'turn' method's options are shown in a dropdown menu, with 'right' and '1 revolution (all the way around)' highlighted.

Jenna's details

properties | **methods** | functions

Jenna move

Jenna **turn**

Jenna roll

Jenna resize

Jenna say

Jenna think

Jenna play sound

Jenna move to

Jenna move toward

Jenna move away from

Jenna orient to

world.my first method No parameters

No variables

Loop 2 times times show complicated version

Jenna.cartwheel

Jenna say I love cartwheels! duration = 2 seconds more...

// Jenna answers her phone and says hello, and then Matt gets c

Jenna.answercellphone

camera turn to face horse1 more...

camera move backward

Matt.RideHorse

camera set direction

camera set left right forward backward

Do in order Do to

amount

1/4 revolution

1/2 revolution

1 revolution (all the way around)

2 revolutions

other...

Loop While For all in order For

As Seen By (Continued)

Click on **more...** next to your **Jenna turn right** command. Select **asSeenBy** and then **horse1**.

The screenshot shows a sequence of commands in an animation software interface. The commands are:

- Jenna answer Cellphone
- camera turn to face horse1 more...
- camera move backward 1.5 meters more...
- Matt.RideHorse
- camera set point of view to horse position more...
- camera set point of view to original position more...
- Jenna turn right 1 revolution more...

The 'Jenna turn right' command is selected, and its 'more...' dropdown menu is open, showing 'asSeenBy' and 'horse1' selected. The 'asSeenBy' dropdown is also open, showing 'the entire horse1' selected. Arrows indicate the selection path from 'more...' to 'asSeenBy' to 'horse1' to 'the entire horse1'.

Now play your animation to see what happens. Jenna should circle around the horse.

As Seen By (Continued)



- Try Changing the object after “as seen by” from the horse to the camera.
- Jenna should disappear off the right side of your screen and reappear on the left side.
- In this picture she is circling off screen, around the horse.

As Seen By Conclusion

- You can use “as seen by” to have an object rotate around another object.
- Whichever object you select first will do the rotating and whichever object you select after “as seen by” will be rotated around.
- Example: Jenna turn left 1 revolution as seen by the horse means that Jenna will rotate around the horse.

Making a Billboard

-We will now show you how to make billboards in Alice.

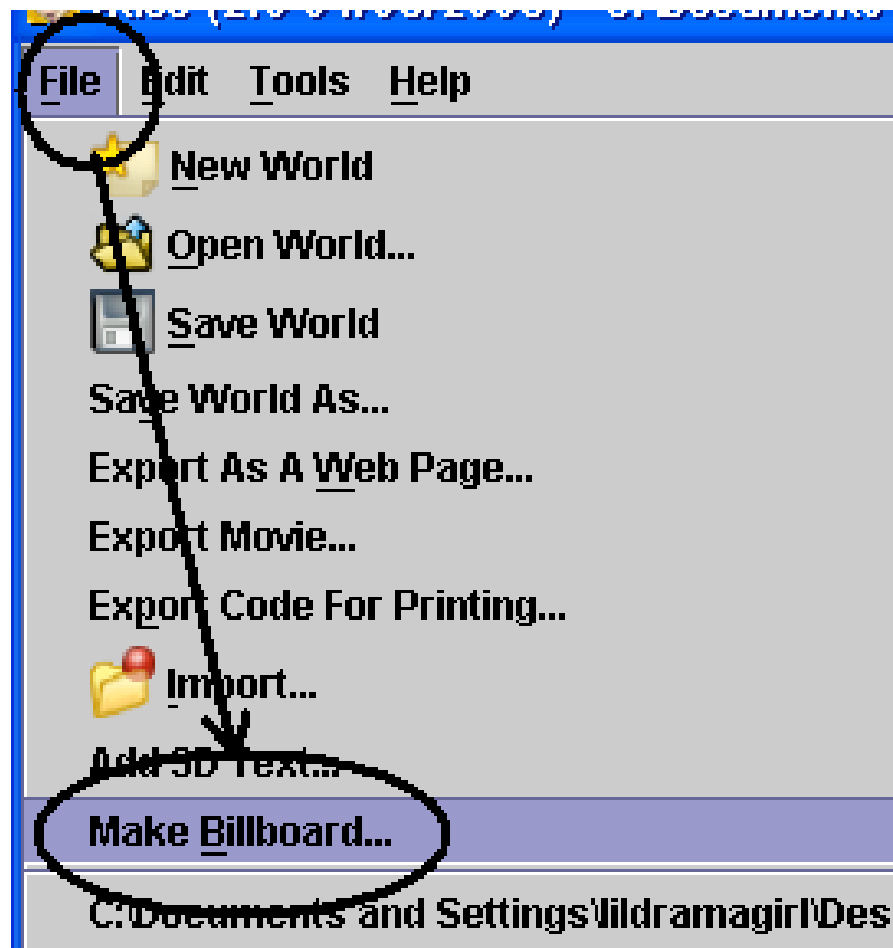
-This is a way you can take pictures that you make in Paint or find on the Internet and use them in your Alice world.

Getting Started

- Go online and find a picture that you like
- In this case it is a stable scene, taken from Google Images.
- Save this image onto your desktop or a folder that is easily accessible.
- Ours is saved on the desktop, as a .jpg



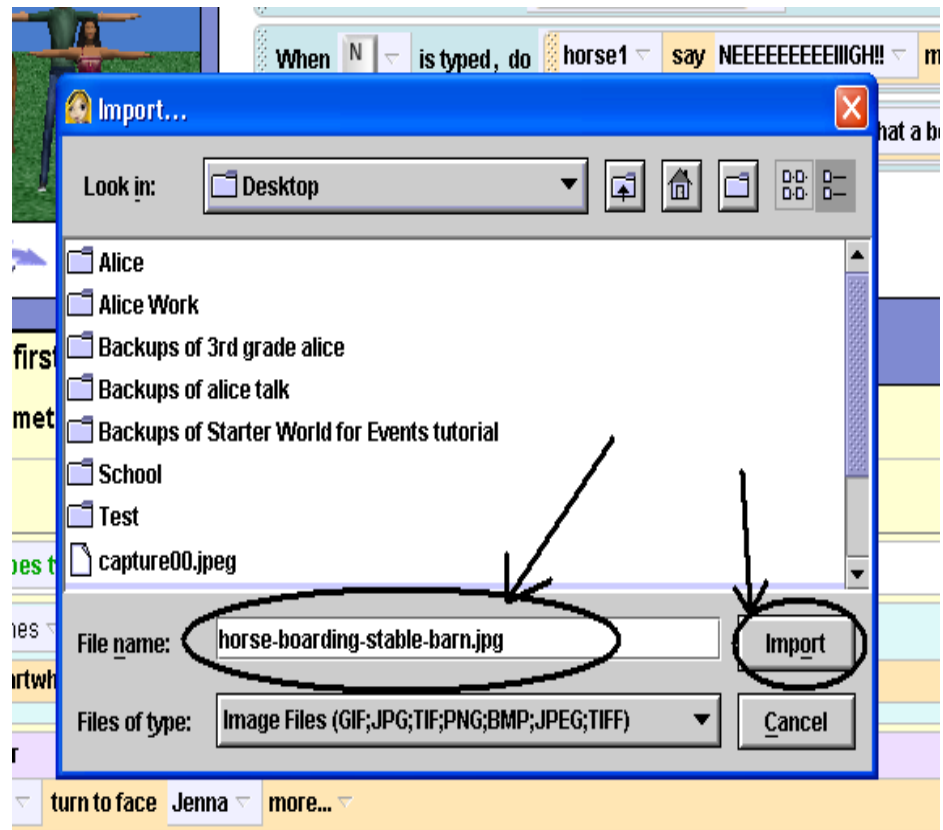
Billboards



- Now click on File and select “make a billboard.”

Billboard

- A pop up box will appear in the middle of your screen.
- Select the .jpg image from your desk top
- Select “import.”



Billboard

- This should import a small picture of your .jpg image into your Alice world (shown behind the people)



Billboards



- Now click on your billboard and stretch it out so that it is big enough to be the background to your animation.

3-D Text

Lets add some 3-D text to our world

This can say “That’s all folks”



- You can change the font below in this square.
- Once you have selected the font it will appear in your world.



3-D Text

- Your text should appear in your world like this.
- You can move it around using the buttons at the right.



- Your screen should look like this.

Coloring Your Text



If you want to color your text you can go to the properties tab of your text and change the color. It should automatically appear on your 3-D text, in your animation.

Animating Your Text

You can animate your text to spin around, and do things just like any other object in your world, simply by writing a method. Let's write a new method to move the 3-D text in and spin it around.

Animating your Text

That's all Folks!!!!.Animate *No parameters*

No variables

That's all Folks!!!! ▾ set isShowing to true ▾ more... ▾

That's all Folks!!!! ▾ move forward ▾ 5 meters ▾ more... ▾

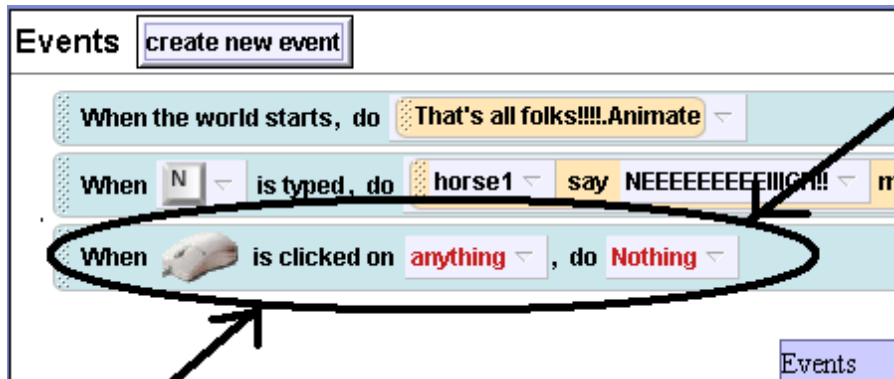
☐ Do together

☐ Loop 3 times ▾ times **show complicated version**

That's all Folks!!!! ▾ turn left ▾ 1 revolution ▾ *duration = 0.5 seconds* ▾ *style = abruptly* ▾ more... ▾

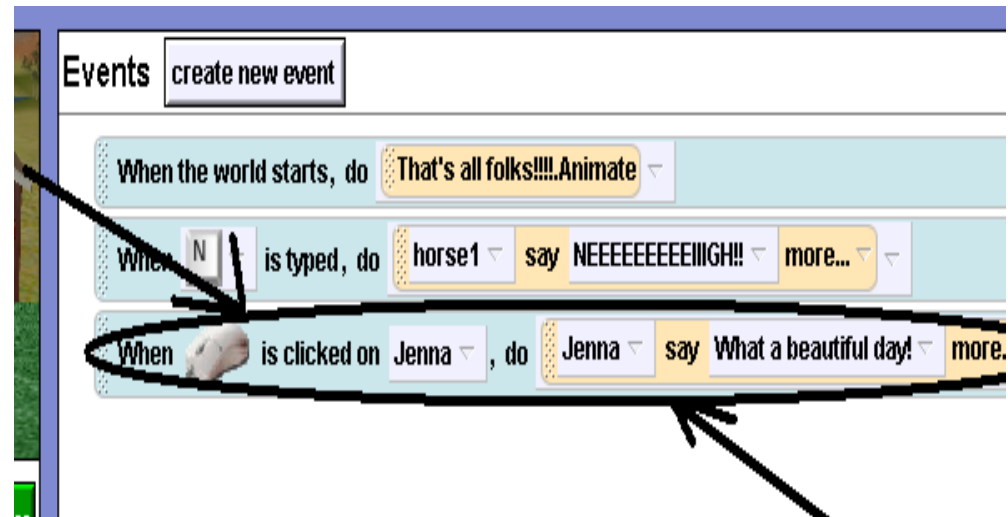
- Write a new class level method that looks like this.
- Call it “animate.”
- Change the event “when the worlds starts” to call the “animate” method and try it out!

New Events

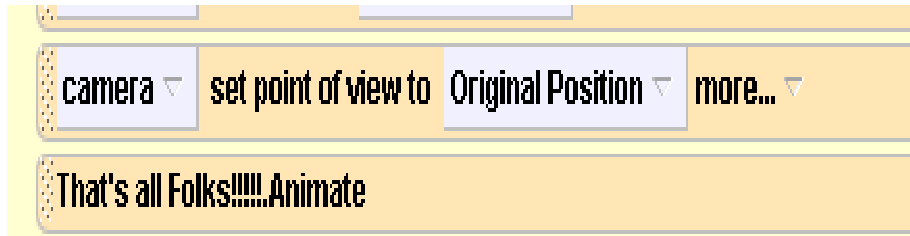


- Select “When the mouse is clicked on something.”
- When the mouse is clicked on Jenna, Jenna say “What a beautiful day”.

- Now lets try a different event.
- Try creating another simple event.



Wrapping up



- Change the event for “when the world starts” back to “myFirstMethod”
- Now your entire world should play as before with the 3-D text animating at the very end

- Finally lets add the 3-D text call to animate to our World.myfirstmethod.
- We need to set the camera back to “Original Position” and then call “That’s all folks.Animate”
- Add this to the very end of your world.myfirstmethod.

Now you should have two events in your animation



- Press “N” and click on Jenna when your story plays
- If you want to learn more about Events, see the “Events Tutorial”.

Congratulations! You now know much of what is possible in the Alice world. Explore on your own, and see what new things you can create and do! Feel free to try our other tutorials to learn more.



Tips & Techniques 1: Special Effects: Text and 2D Graphic Images

While it is beyond the scope of this text to cover all the special capabilities of Alice, we would like to show you some particular features of Alice as we work through examples. At the end of each chapter, Tips and Techniques sections will present fun ways to build great animations with special effects. While not essential to learning to program, Tips & Techniques sections are important to read because some of these techniques are used in example worlds. Topic subheadings in **red** are used in text examples, subheadings in **green** are used in exercises. Other topics with subheadings in **black** are provided as a guide for those who want to learn more about animation with Alice. Additional details of using Alice are also provided in Appendix B, where you can learn how to search the gallery and how to export a world to the web. The Tips & Techniques sections along with Appendix A and Appendix B can be considered a mini-User's Guide for Alice.

An important part of an animation is communicating information to the person viewing the animation (the **user**). Text, sound, and graphic images help you communicate. The following sections show how to add text and graphic images to your world.

Text

The *Say* instruction (introduced in Getting Started, Appendix A) is one way to create text in a scene. The text created by a *Say* instruction appears inside a comic-book-style bubble. 3D text is another kind of text display. To add a 3D text object to a world, click on the Create 3D text folder in the local gallery, as seen in Figure T-1-1.

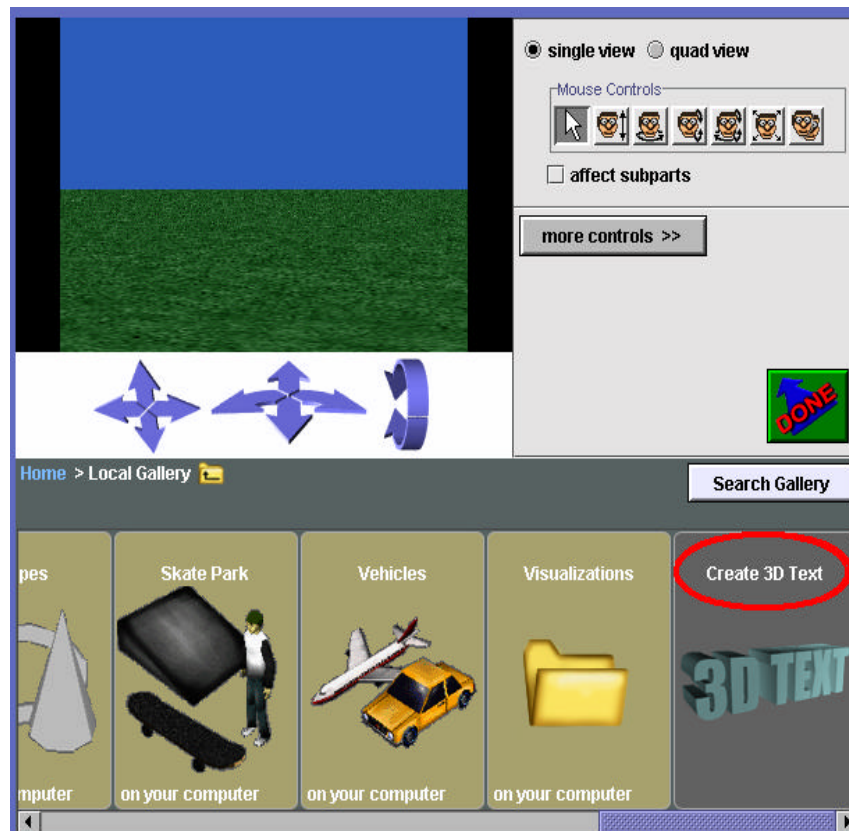


Figure T-1-1. 3D text in the local gallery

A text dialog box pops up for entering text, as in Figure T-1-2. The dialog box allows font, bold, and italic selections, and a Text box where words can be typed.



Figure T-1-2. A text dialog Box

When the Okay button is clicked, Alice adds a text object to the world and an entry for the object in the object tree. The name of the object is the same as the text string displayed, as can be seen in Figure T-1-3.

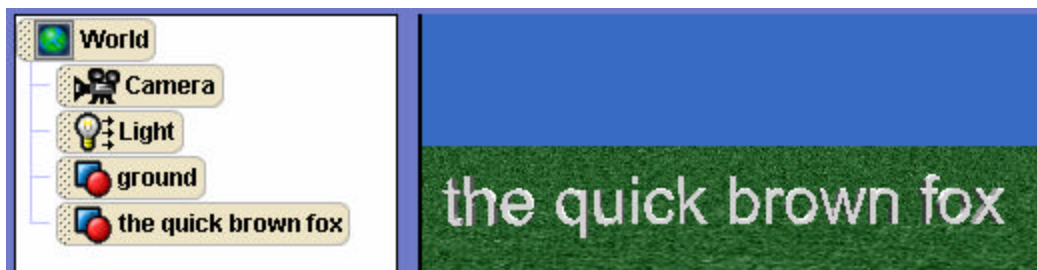


Figure T-1-3. The text object is added to scene and object tree

The text object can be positioned using mouse controls in the same way as any other object. To modify the text in the object string, click on the text in the properties list of Details area. Then, enter a new string of text in the popup dialog box, illustrated in Figure T-1-4.

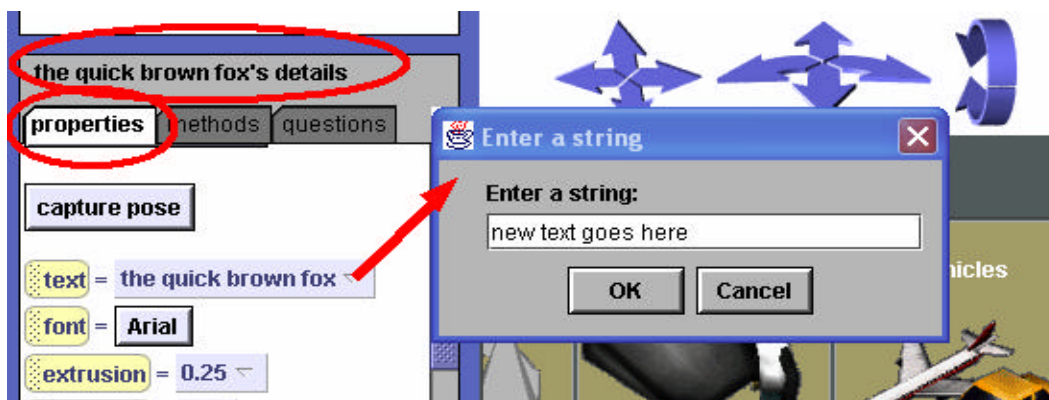


Figure T-1-4. Modifying text

Note that modifying the string in the text object does not modify the name of the object. The name is still as it was when the text object was originally created, as seen in Figure T-1-5.

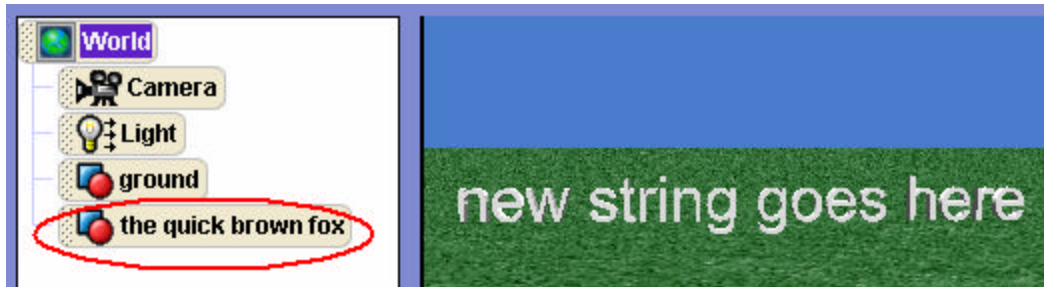


Figure T-1-5. The name of text object remains the same

Graphic images (billboards)

Although Alice is a 3D system, it is possible to display flat 2D images in a scene. Flat 2D images can be created in any paint tool and saved in GIF, JPG, or TIF format. To add a 2D image (Alice calls it a **billboard**) to your world, select Make Billboard from the File menu, as seen in Figure T-1-6. In the selection dialog box, navigate to the stored image and then click the Import button.

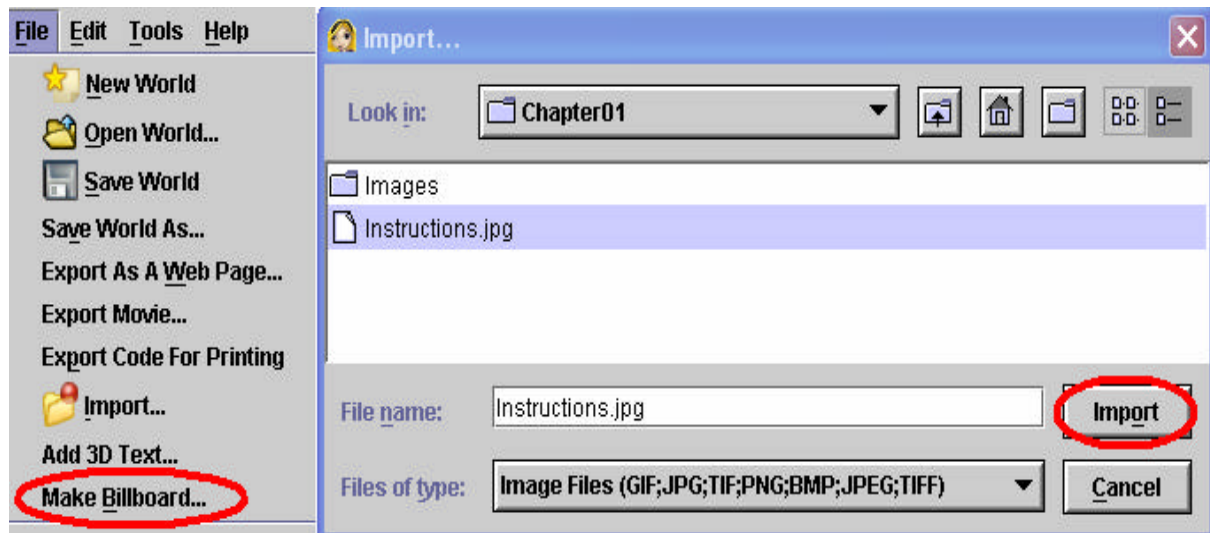


Figure T-1-6. Importing a billboard

Alice will add the flat image to the world. The billboard in Figure T-1-7 illustrates one of the uses of billboards -- providing information to the user about how to play a game or simulation. In this example, the billboard provides instructions for how to use the keyboard to control the motion of an object. (Keyboard control examples are used in Chapter 5.)

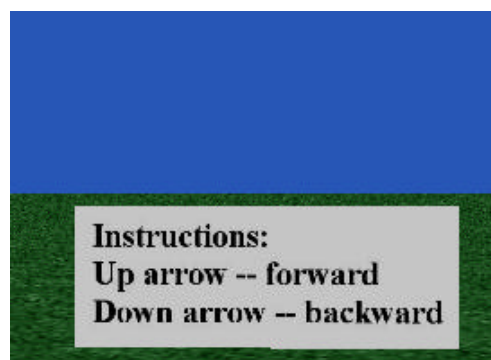


Figure T-1-7. A billboard to provide information

Video Export

Alice 2.2 implements video export capability that allows you to create a QuickTime video of your “movie” worlds. Note that “interactive” worlds work with events that may interrupt the animation process. Therefore, interactive worlds are not ideal for video export.

Short Tour for exporting an Alice world as a QuickTime® video

1. **From the File menu, select Export Video.**

A video capture window opens.

2. **In the video capture window, click Record.**

In the upper right of the window, a red dot blinks and a timer displays progress

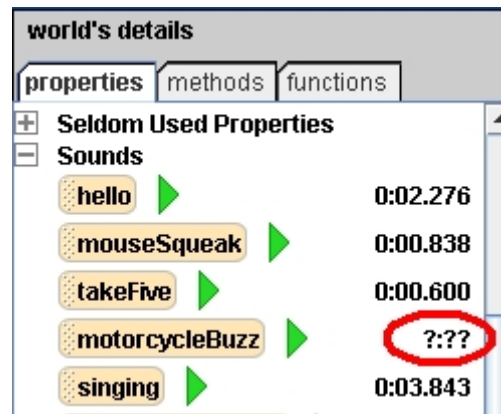
3. **When done recording, click Stop Recording.**

4. **Enter a name for the video in the File name box and then click Export Video.**

If all has gone well, the video file is now located in the same folder as your Alice world and can be viewed in a QuickTime player or posted on a web page for others to view.

Hints on Loading an Alice 2.0 world into Alice 2.2

If you open an Alice 2 world in Alice 2.2 and get a “media.PlaybackEngine” error, then it is likely that at least one of your sound files is not being loaded. To find out which sound file is not being loaded, check the list of imported sounds in the details panel. See example here:



If a sound file has not loaded properly, Alice displays “?:??” as the number of seconds. In this example, *motorcycleBuzz* is not loading properly when the world is opened. The sound file does not load because its format is not recognized by Alice 2.2’s video sound playback engine. To fix this problem, open the original sound file in a sound editor and export the sound as a WAV file to reformat. Now, import the newly formatted file into your Alice world (replacing the previous version).

Note: Alice is not a sound file editor. Many different versions of sound file editors can be found on the Internet. For example, Audacity (freely downloadable from the Internet) is a sound editor that might be a used for reformatting sound files.

Troubleshooting video export in Alice 2.2

Symptom: “I click the Record button and Alice seems to “freeze up” or seems to be taking a VERY long time to complete the recording.”

Likely cause: As the world is being recorded, temporary screen captures are created and stored in a folder named “frames.” You could be trying to save the video on a flash drive (or memory key) that is out of space. Or, the folder you want to use could be a “read only” folder on your computer.

Solution: Close the video capture window and then save the world to a different folder in a different location on your computer. With the world in a different location, start over to create the video.

Another likely cause: As the world is being recorded, a sound file cannot be accessed. This may be because the sound file is not a readable format (see hints on reformatting a sound file, above in this document) or your computer’s operating system has not closed the sound file (from a previous run of the world.)

Solution: Exit Alice, restart and try again (this will close any open sound files). If this doesn’t work, then open the original sound file in a sound editor and export the sound as a WAV file to reformat. Now, import the newly formatted file into your Alice world (replacing the previous version).

Symptom: “I want to start over to export the video but I keep getting parts of the previous recording.”

Likely cause: As the world is being recorded, temporary screen captures are created and stored in a folder named “frames.” The temporary frames may still be there if you start over.

Solution: Click the clear button to remove all the frames before starting over.

Symptom: “I want to start over to export the video and overwrite the previous recording but I get a message saying the previous video file is in use.”

Likely cause: The previous video file is still “open” either in Alice 2.2 or in QuickTime.

Solution: Close Alice 2.2 and QuickTime and then restart Alice 2.2, load the world, and start over.

Symptom: “I uploaded my video to YouTube and discovered that on YouTube the video seems to have lost most of its sound.”

Likely cause: Any video uploaded to YouTube is automatically converted to a Flash® format (unless it was already in Flash format). Alice videos having only one sound file generally convert well. But, if the Alice video has multiple sound files it may or may not convert properly.

Solution: We are still working on a solution to this problem. A couple of Alice users

have suggested using a sound editor (such as Audacity) to create a single sound track from the multiple sound files.